Video Game Development

CURRICULUM

Training Manual

TABLE OF CONTENTS

Certification Module I

Game Engines, Programs, Basic Commands

Certification I Worksheet

Teacher's Version

The game engine used in this certification is called Dark Basic .
(found on Certification page)

The folder that the certification files are found under is

myproj \ isl
(found in Introduction to Dark Basic video)

Show two different series of steps that did give you a glass of chocolate milk.

glass	glass
milk	chocolate
chocolate	milk
stir	stir
straw	straw

(found in Chocolate Milk Simulator)

Program languages have grammar just like English, Spanish, and Swahili. Programmers refer to the grammar of programming language as
syntax .
(found on Certification page)

Some common programming languages are:

Java, C++, Basic, C, HTML

(found on Certification page)

Remark statements, or REM statements, are used to do what?

REM statements allow a programmer to insert text into the program that can be read while viewing the program code, but is not used by the program or seen by the user.

(found in Remark Statements video)

Certification I Worksheet

Teacher's Version

List two reasons why program structure is important when designing computer game programs.

Program structure helps to organize the parts of a program. It makes it easier for other programmers. It's a common map of the program. Different programming languages require the program to be in a certain sequence or order.

(found on Certification page and in Program Structure video)

When editing a program with the editor, it is similar to using A .

(found in Editing Program Code video)

A. A word processor
B. A calculator
C. A pad of paper and ruler

The Cube's scale changed the Cube's C .

(found in Changing the Cube's Scale video)

A. Color
B. Texture
C. Size

Examples of two different commands you used with the CLI (Command Line Interface) to write text onto the screen.

print "hello"

center text 320,240 "hello"

(found on Certification page and in Print and Text Using the CLI video)

A DO LOOP is used to A .

(found in Do Loops video)

A. Do a series of actions until a condition inside the loop causes the program to move out of the loop.
B. Direct Objects through a loop structure
C. Make a decision

The command *set cursor 0,50* would have the text placed A .

(found in Problem with the Program video)

A. On the left edge of the screen and 50 pixels down from the top
B. On the right edge of the screen and 50 pixels up from the bottom
C. Horizontally centered at 0 with a length of 50

Game Engines and Programming

Game engines are special software programs that help programmers create games. If you have ever heard of the game DOOM (© id Software), then you also know one of earliest and most successful games to be created using a game engine. Before games like Doom and Wolfenstein, the programmers built the games from scratch. Id Software proved the value of having a solid game engine to use, and the concept of the game engine helped fuel the success of the FPS (First Person Shooter) games, which paved the way for a new approach to creating video games.

So what makes a game engine a game engine? Let's look at a race car as a model. You can take the engine out of the car and build a new body around it. The outside will look like a new car, but what "drives" the car is the same "engine." Just like a car's engine is not specific to the car, a game engine is not specific to a game. To programmers, an engine is the non-game-specific software that the game uses to run.

Typically, a game engine will include a *renderer* (which presents the player's point of view on the screen), a *culler* (which removes objects that cannot be seen at that moment of play), a *3D world generator* (which keeps track of where objects are in the "game's world"), and many other bits to make the development of a game easier.

Let's take a quick tour the game engine you will be using in your development work.

VIDEO: Introduction to Dark Basic (3:54)

WORKSHEET

The game engine used in this certification is called *Dark Basic*.

The folder that the certification files are found under is *myproj\isl*

Programs Are...

You can think of a program as a list of things to do, in sequence. Since computers are essentially stupid, you have to tell the computer everything it needs to know, including when and how it should do it.

If a program is nothing more than a series of steps in a specific sequence, then you and I perform "programs" each day. Brushing your teeth, washing your hair, making a bowl of cereal are all examples of regular activities that you do that can be described as a series of actions, or a program.

Use the animation below, and try your hand at the sequence of steps necessary to make a glass of chocolate milk.

ANIMATION: Chocolate Milk Program Simulator

WORKSHEET

Show two different series of steps that did give you a glass of chocolate milk.
glass, milk, chocolate, stir, straw
glass, chocolate, milk, stir, straw

Loading in a Game into the Game Engine

VIDEO: Load and Run a Program (0:40)

To start up Dark Basic, look for this icon on your computer's desktop:

Your Action:
Start up Dark Basic. Load in the game file ***certification1_1.dba***. Run the game by selecting RUN and then EXECUTE from the top menu.

The game you just loaded and used was a simple program. The program waited to see what key you pressed and then reacted by showing you the cube moving on the screen. As a game goes, the Cube Game really isn't much of a game, but it is a simple example of what a game program does. It gets input from the player, combines it with information from the game program, and displays what is happening in the "game world".

Writing a Program

Writing a program is similar to writing a play. A play has a beginning, middle, and ending; so does a program. Just like a play, a program uses a language. This language has the correct grammar -- also called *syntax* -- that must be used so the actors and those watching the play can understand correctly what the playwright wants. A good playwright carefully writes what each actor is to do and say, so that all the action and dialog of the play flows smoothly.

WORKSHEET

Program languages have grammar just like English, Spanish, and Swahili. Programmers refer to the grammar of programming language as *syntax*.

Programs are written in a specific language, just like people in Germany speak German, and people in Portugal speak Portuguese. Different programs are written in different languages, and programmers typically become proficient in a few different languages. Deciding the language to use depends on a number of different factors. On what hardware will the program be used? What type of program is it? Will the program be used over the internet? Depending on these factors and a few more, the program may be written in any number of different program languages. A few common languages in use today are Basic, Java, C++, C, and HTML.

WORKSHEET

Some common programming languages are: *Java, C++, Basic, C, HTML*

WATCH VIDEO: Remark Statements (1:51)

WORKSHEET

Remark statements, or REM statements, are used to do what?

REM statements allow a programmer to insert text into the program that can be read while viewing the program code, but is not used by the program or seen by the user.

Your Action:

Load in the game file ***certification1_1.dba*** and see how REM statements are used to document the program.

certification1_1.dba

REM ***

⬆ A REM statement tells the compiler to ignore this line of code. It is for the reader's benefit only.

RemStart ⬅ This command marks the beginning of a block of "remmed-out" code

```
***
*** TITLE - Introduction to DarkBASIC
*** VERSION - 1.1 LAST UPDATED - 1.1.2000
*** DEVELOPER - Jason Holm
*** COPYRIGHT - Ingenious Student Labs
*** DATE CREATED - 1.1.2000
***
```

⬆ All the lines between a RemStart and RemEnd will be ignored (the *'s are just for looks)

RemEnd ⬅ This command marks the end of a block of "remmed-out" code

```
REM *** START SYSTEM SETUP SECTION

sync on : sync rate 30
autocam on
hide mouse

REM *** STOP SYSTEM SETUP SECTION
REM ***************************************

REM ***************************************
REM *** START INTRO SECTION
REM *** INTRO SECTION HEADER

REM SET VARIABLES
REM SCREEN DISPLAY
cls 0 : center text 320,240,"HIT ANY KEY TO BEGIN"
REM LOAD SOUNDS
REM SPECIAL EFFECTS
REM REFRESH SCREEN
sync

REM *** INTRO SECTION LOOP

do
   REM CONTROL INPUT
   x=scancode() : if Keystate(x)=1 then goto MainSection
   REM REFRESH SCREEN
   sync
loop

REM *** END INTRO SECTION
REM ***************************************
```

```
REM ****************************************
REM *** START MAIN SECTION
REM *** MAIN SECTION HEADER

MainSection:
REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls 0 : backdrop on
REM LOAD TEXTURES
REM LOAD SOUNDS
REM OBJECT CREATION
make object cube 1,100
REM LOAD MODELS
REM SET CAMERA
REM REFRESH SCREEN
sync

REM *** MAIN SECTION LOOP

do
   REM SPECIAL EFFECTS
   REM OBJECT ORIENTATIONS
   1aY# = object angle Y(1)
   REM CAMERA ORIENTATIONS
   REM LIVE SCREEN DISPLAY
   center text 320,440,"USE ARROW KEYS TO MOVE    |    PRESS 'Q' TO
QUIT"
   REM CONTROL INPUT
   if Upkey()=1 then move object 1,10
   if Leftkey()=1 then 1aY# = wrapvalue(1aY#-5)
   if Rightkey()=1 then 1aY# = wrapvalue(1aY#+5)
   if Inkey$()="q"
      delete object 1 : backdrop off : goto EndSection
   endif
   REM TRANSFORM OBJECTS
   REM MOVE OBJECTS
   Yrotate object 1,1aY#
   REM CHECK FOR COLLISION
   REM REFRESH SCREEN
   sync
loop

REM *** STOP MAIN SECTION
REM ****************************************
```

```
REM ****************************************
REM *** START END SECTION
REM *** END SECTION HEADER

EndSection:
REM SCREEN DISPLAY
cls 0 : center text 320,240,"PLAY AGAIN [Y/N]?"
REM LOAD SOUNDS
REM REFRESH SCREEN
sync

REM *** END SECTION LOOP

do
   REM CONTROL INPUT
   if Inkey$()="y" then goto MainSection
   if Inkey$()="n"
      cls : end
   endif
   REM REFRESH SCREEN
   Sync
loop

end
REM *** STOP END SECTION
REM ****************************************
```

Program Structure

Just like a well-written play, programs are well organized, and this organization may be called a structure. A structure is important in a number of ways. Different programming languages require different parts of a program to be in a specific sequence. A program structure gives programmers a common "map", which can be very important when you have many programmers working on a large project together. A typical game program structure may look like this:

Introduction Section or **Beginning** -- The game needs to prepare itself

Title Section -- Tells who wrote the program, version information, copyright, and the like
System Setup Section -- Load special drivers, set system settings like sync rate, sound card, mouse or controller settings
Introduction Screens -- Display different screens before the game begins -- starting title text, game options screens, level intros

Main Section or **Middle** -- Where the real action in the game takes place

Main Header -- Declare variables, load in sounds, textures, models, set starting camera view
Main Loop -- Takes the users input and displays the game action on the screen

End Section or **Ending** -- What happens when the game is over -- Display final score screen, may offer restart game option or shuts program down and clears memory

Now let's look at an example of structure in a game program.

VIDEO: Program Structure (4:00)

WORKSHEET
List two reasons why program structure is important when designing computer game programs.
Program structure helps to organize the parts of a program. It makes it easier for other programmers. It's a common map of the program. Different programming languages require the program to be in a certain sequence or order.

Before we practice editing programs, check out this video that talks about editing code.

VIDEO: Editing Program Code (1:58)

WORKSHEET
When editing a program with the editor, it is similar to using *A: A word processor*.

Now watch this video demonstrating how to edit the size of a cube in the game program.

VIDEO: Changing the Cube's Size (2:47)

Your Action:
Load ***certification1_2.dba*** and change the cube size from 100 to 10, 50, and 500.

WORKSHEET
Changing the cube's scale affected its *C: Size*.

certification1_2.dba

```
REM ****************************************
RemStart
    ***
    *** TITLE - Program Structure
    *** VERSION - 1.2 LAST UPDATED - 1.1.2000
    *** DEVELOPER - Jason Holm
    *** COPYRIGHT - Ingenious Student Labs
    *** DATE CREATED - 1.1.2000
    ***
RemEnd
REM *** START SYSTEM SETUP SECTION

sync on : sync rate 30
autocam off
hide mouse

REM *** STOP SYSTEM SETUP SECTION
REM ****************************************

REM ****************************************
REM *** START INTRO SECTION
REM *** INTRO SECTION HEADER

REM SET VARIABLES
REM SCREEN DISPLAY
cls 0 : center text 320,240,"HIT ANY KEY TO BEGIN"
REM LOAD SOUNDS
REM SPECIAL EFFECTS
REM REFRESH SCREEN
sync

REM *** INTRO SECTION LOOP

do
   REM CONTROL INPUT
   x=scancode() : if Keystate(x)=1 then goto MainSection
   REM REFRESH SCREEN
   sync
loop

REM *** END INTRO SECTION
REM ****************************************
```

```
REM ****************************************
REM *** START MAIN SECTION
REM *** MAIN SECTION HEADER

MainSection:
REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls 0 : backdrop on
REM LOAD TEXTURES
REM LOAD SOUNDS
REM OBJECT CREATION
make object cube 1,100 ⬅ Here is where you will change the cube's size
REM LOAD MODELS
REM SET CAMERA
position camera 0,100,-250
point camera 0,0,0
REM REFRESH SCREEN
sync

REM *** MAIN SECTION LOOP

do
   REM SPECIAL EFFECTS
   REM OBJECT ORIENTATIONS
   1aY# = object angle Y(1)
   REM CAMERA ORIENTATIONS
   REM LIVE SCREEN DISPLAY
   center text 320,440,"USE ARROW KEYS TO MOVE   |   PRESS 'Q' TO
QUIT"
   REM CONTROL INPUT
   if Upkey()=1 then move object 1,10
   if Leftkey()=1 then 1aY# = wrapvalue(1aY#-5)
   if Rightkey()=1 then 1aY# = wrapvalue(1aY#+5)
   if Inkey$()="q"
      delete object 1 : backdrop off : goto EndSection
   endif
   REM TRANSFORM OBJECTS
   REM MOVE OBJECTS
   Yrotate object 1,1aY#
   REM CHECK FOR COLLISION
   REM REFRESH SCREEN
   sync
loop

REM *** STOP MAIN SECTION
REM ****************************************
```

```
REM ****************************************
REM *** START END SECTION
REM *** END SECTION HEADER

EndSection:
REM SCREEN DISPLAY
cls 0 : center text 320,240,"PLAY AGAIN [Y/N]?"
REM LOAD SOUNDS
REM REFRESH SCREEN
sync

REM *** END SECTION LOOP

do
   REM CONTROL INPUT
   if Inkey$()="y" then goto MainSection
   if Inkey$()="n"
      cls : end
   endif
   REM REFRESH SCREEN
   sync
loop

end
REM *** STOP END SECTION
REM ****************************************
```

Basic Commands

Printing and Text

One of the first commands any programmer learns is to print text to the screen. This may seem quite simple, and it is, but it is also the first method your program will have to communicate to the user. Games need to communicate to the user to give scores, control or game options, or just to tell you that you have died a horrible death at the hands of a smiling monster. Fortunately, the command to print text to the screen is quite simple and easy to practice.

There are three ways to get text onto the screen:

1. The PRINT command will print text or the value of a variable on to the screen at the top right (or 0,0) position.

2. A text command will allow you to print text at a defined position on the screen. In the picture below, you will see the text "Your Score is" printed at a position of 320,240.

3. The third way is to create the text as part of the background or images that are shown in the game.

VIDEO: Print and Text Using the CLI (2:13)

WORKSHEET

Examples of two different commands you used with the CLI (Command Line Interface) to write text onto the screen.

print "hello"

center text 320,240 "hello"

Your Action:

Start the game engine and click on the CLI button at the top right to activate the Command Line Interface. At the bottom of the screen right below the line that begins "*>>> YOU CAN PRESS F12…"* type the following:

print "hello"

Then press enter -- You should see *hello* appear in the upper left corner of the screen.

On the next line type:

center text 320,240, "Your Score is"

Then press enter and you will see the text appear in the center of the screen. Your screen should look like the picture shown here.

Now print your name onto the screen using both the PRINT and CENTER TEXT commands. What happens when you change the numbers on ***center text 320,240*** to 100,200 or 500,50?

Now let's work with some text inside a program.

Your Action:
Load in ***certification1_2.dba*** and change the text for "HIT ANY KEY TO BEGIN" to have the program tell ***you*** to hit any key. For example, if your name is Michael, have the program print, "MICHAEL, HIT ANY KEY TO BEGIN". You will find the text in the Intro Section Header under the line REM SCREEN DISPLAY.

certification1_2.dba

```
REM ****************************************
RemStart
    ***
    *** TITLE - Program Structure
    *** VERSION - 1.2 LAST UPDATED - 1.1.2000
    *** DEVELOPER - Jason Holm
    *** COPYRIGHT - Ingenious Student Labs
    *** DATE CREATED - 1.1.2000
    ***
RemEnd
REM *** START SYSTEM SETUP SECTION

sync on : sync rate 30
autocam off
hide mouse

REM *** STOP SYSTEM SETUP SECTION
REM ****************************************

REM ****************************************
REM *** START INTRO SECTION
REM *** INTRO SECTION HEADER

REM SET VARIABLES

REM SCREEN DISPLAY ⬅
cls 0 : center text 320,240,"HIT ANY KEY TO BEGIN"
        ⬆ Here is where you will change the text printed to the screen

REM LOAD SOUNDS
REM SPECIAL EFFECTS
REM REFRESH SCREEN
sync

...
```

There are other ways to have text appear on the screen. Professional game designers will use backgrounds to make the screens seem more interesting.

VIDEO: Using a Background (2:39)

Your Action:

Load in ***certification1_3.dba***. As you saw in the video, go under the REM SCREEN DISPLAY section and change

useBackground = 0

to

useBackground = 1

Run the program to see the effect of having a background for your screen.

certification1_3.dba

```
REM ****************************************
RemStart
    ***
    *** TITLE - Print and Text Commands / Background Color and Images
    *** VERSION - 1.3 LAST UPDATED - 1.1.2000
    *** DEVELOPER - Jason Holm
    *** COPYRIGHT - Ingenious Student Labs
    *** DATE CREATED - 1.1.2000
    ***
RemEnd
REM *** START SYSTEM SETUP SECTION

sync on : sync rate 30
autocam on
hide mouse

REM *** STOP SYSTEM SETUP SECTION
REM ****************************************

REM ****************************************
REM *** START INTRO SECTION
REM *** INTRO SECTION HEADER

REM SET VARIABLES
REM SCREEN DISPLAY
cls rgb(0,0,0)
   RemStart
   Clear screen and set background color
   [rgb(0 to 255 for red, 0 to 255 for green, 0 to 255 for blue)]
   Default color is BLACK [0 or rgb(0,0,0)]
   or last set background color
   RemEnd

useBackground = 0 ⬅ Change this variable from 0 to 1 to see the difference

if useBackground = 1 then load bitmap "images/cubetitle.bmp"

if useBackground = 0
   ink rgb(255,255,255),rgb(0,150,0)
   RemStart
   Set text color [FOREGROUND, BACKGROUND]
   Default colors are WHITE FOREGROUND [rgb(0,0,0)]
   and BLACK BACKGROUND [0 or rgb(0,0,0)]
   or last set colors
   RemEnd
   set text font "arial" : Rem Set text font
   set text size 36 : Rem Set text size
   set text to bold : Rem Set text style
[normal,bold,italic,bolditalic]
   set text opaque : Rem Set text transparency [opaque,transparent]
```

```
   Rem Print text centered at a specific location on screen
   center text 320,240," THE CUBE GAME "
   sync

   Rem Adjust text attributes
   set text font "times" : set text size 12
   set text to italic : set text transparent
   Rem Print text beginning at a specific location on screen
   text 320,276,"DEVELOPED BY JASON HOLM"
   sync

   set text font "times" : set text size 20
   set text to bold  : set text transparent
   center text 320,300,"HIT ANY KEY TO BEGIN"
   sync
endif

set cursor 0,0
   RemStart
   Prepare to type in a specific place on screen [X,Y]
   Default cursor location is 0,0
   or the next line after the last printed line
   RemEnd
ink rgb(255,255,255),0
print "Copyright (c) Ingenious Student Labs"
   RemStart
   NOTE: The 'print' function does not utilize
   centering, font, size, style, background color or transparency
   RemEnd
sync
REM LOAD SOUNDS
REM SPECIAL EFFECTS
REM REFRESH SCREEN
sync

REM *** INTRO SECTION LOOP

do
   REM SCREEN DISPLAY
   REM CONTROL INPUT
   x=scancode() : if Keystate(x)=1 then goto MainSection
   REM REFRESH SCREEN
   sync
loop

REM *** END INTRO SECTION
REM ***************************************
```

```
REM ****************************************
REM *** START MAIN SECTION
REM *** MAIN SECTION HEADER

MainSection:
REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls 0 : backdrop on
REM LOAD TEXTURES
REM LOAD SOUNDS
REM OBJECT CREATION
make object cube 1,100
REM LOAD MODELS
REM SET CAMERA
REM REFRESH SCREEN
sync

REM *** MAIN SECTION LOOP

do
   REM SPECIAL EFFECTS
   REM OBJECT ORIENTATIONS
   1aY# = object angle Y(1)
   REM CAMERA ORIENTATIONS
   REM LIVE SCREEN DISPLAY
   set text font "times" : set text size 16
   set text to bold  : set text transparent
   center text 320,440,"USE ARROW KEYS TO MOVE   |   PRESS 'Q' TO
QUIT"
   REM CONTROL INPUT
   if Upkey()=1 then move object 1,10
   if Leftkey()=1 then 1aY# = wrapvalue(1aY#-5)
   if Rightkey()=1 then 1aY# = wrapvalue(1aY#+5)
   if Inkey$()="q"
      delete object 1 : backdrop off : goto EndSection
   endif
   REM TRANSFORM OBJECTS
   REM MOVE OBJECTS
   Yrotate object 1,1aY#
   REM CHECK FOR COLLISION
   REM REFRESH SCREEN
   sync
loop

REM *** STOP MAIN SECTION
REM ****************************************
```

```
REM ****************************************
REM *** START END SECTION
REM *** END SECTION HEADER

EndSection:
REM SCREEN DISPLAY
cls 0
set text font "times" : set text size 20
set text to bold  : set text transparent
center text 320,240,"PLAY AGAIN [Y/N]?"
REM LOAD SOUNDS
REM REFRESH SCREEN
sync

REM *** END SECTION LOOP

do
   REM CONTROL INPUT
   if Inkey$()="y" then goto MainSection
   if Inkey$()="n"
      cls : end
   endif
   REM REFRESH SCREEN
   sync
loop

end
REM *** STOP END SECTION
REM ****************************************
```

Do Loops

Loops are an important part of programming and allow the program do many different things. There are many different kinds of loops like Do, While, and the For/Next. The first loop we will learn is the Do Loop. You may have seen movies where a computer gets into an "Infinite Loop" that then causes the computer to lock up and then the world blows up – or something like that. Well, a Do Loop will continue to do the Do until you force it out of the loop.

ANIMATION: Pit Stop

The Do Loop looks like this:

```
do
  List of things you want to happen while the loop is running
loop
```

Now let's go look at Do Loops inside a program.

VIDEO: Do Loops (4:20)

WORKSHEET

A DO LOOP is used to A: Do a series of actions until a condition inside the loop causes the program to move out of the loop.

Your Action:

Load the ***certification1_4.dba*** program. Under the INTRO SECTION LOOP section, change

```
if Keystate(x)=1 then goto MainSection
```

to

```
REM if Keystate(x)=1 then goto MainSection
```

certification1_4.dba

```
REM ****************************************
RemStart
    ***
    *** TITLE - Do Loops
    *** VERSION - 1.4 LAST UPDATED - 1.1.2000
    *** DEVELOPER - Jason Holm
    *** COPYRIGHT - Ingenious Student Labs
    *** DATE CREATED - 1.1.2000
    ***
RemEnd
REM *** START SYSTEM SETUP SECTION

sync on : sync rate 30
autocam on
hide mouse

REM *** STOP SYSTEM SETUP SECTION
REM ****************************************

REM ****************************************
REM *** START INTRO SECTION
REM *** INTRO SECTION HEADER

REM SET VARIABLES
REM SCREEN DISPLAY
cls
load bitmap "images/cubetitle.bmp"
print "Copyright (c) Ingenious Student Labs"
REM LOAD SOUNDS
REM SPECIAL EFFECTS
REM REFRESH SCREEN
sync

REM *** INTRO SECTION LOOP

do
   REM SCREEN DISPLAY
   set cursor 0,40
      RemStart
      What happens if we REM the SET CURSOR command
      in a loop?
      RemEnd
   print "CUBES ARE FUN!"
```

```
   REM CONTROL INPUT
   x=scancode()
   if Keystate(x)=1 then goto MainSection
   ⬆ Add a REM command in front of this line to see the difference
      RemStart
      Alternately, you could use
         if Keystate(x)=1 then exit
      This tells the program to exit this loop
      and continue to the next set of instructions
      RemEnd
   REM REFRESH SCREEN
   sync
loop

REM *** END INTRO SECTION
REM ***************************************

REM ***************************************
REM *** START MAIN SECTION
REM *** MAIN SECTION HEADER

MainSection:
REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls 0 : backdrop on
REM LOAD TEXTURES
REM LOAD SOUNDS
REM OBJECT CREATION
make object cube 1,100
REM LOAD MODELS
REM SET CAMERA
REM REFRESH SCREEN
sync

REM *** MAIN SECTION LOOP

do
   REM SPECIAL EFFECTS
   REM OBJECT ORIENTATIONS
   1aY# = object angle Y(1)
   REM CAMERA ORIENTATIONS
   REM LIVE SCREEN DISPLAY
   ink rgb(255,255,255),0 : set text font "times" : set text size 16
: set text to bold : set text transparent
   center text 320,440,"USE ARROW KEYS TO MOVE   |   PRESS 'Q' TO
QUIT"
   set cursor 0,0
      RemStart
      What happens if we REM the SET CURSOR command
      in a loop that uses a backdrop?
      RemEnd
   print "CUBES ARE FUN!"
```

```
   REM CONTROL INPUT
   if Upkey()=1 then move object 1,10
   if Leftkey()=1 then 1aY# = wrapvalue(1aY#-5)
   if Rightkey()=1 then 1aY# = wrapvalue(1aY#+5)
   if Inkey$()="q"
      delete object 1 : backdrop off : goto EndSection
   endif
   REM TRANSFORM OBJECTS
   REM MOVE OBJECTS
   Yrotate object 1,1aY#
   REM CHECK FOR COLLISION
   REM REFRESH SCREEN
   sync
loop

REM *** STOP MAIN SECTION
REM ***************************************

REM ***************************************
REM *** START END SECTION
REM *** END SECTION HEADER

EndSection:
REM SCREEN DISPLAY
cls 0
ink rgb(255,255,255),0 : set text font "times" : set text size 20 :
set text to bold : set text transparent
center text 320,240,"PLAY AGAIN [Y/N]?"
REM LOAD SOUNDS
REM REFRESH SCREEN
sync

REM *** END SECTION LOOP

do
   REM CONTROL INPUT
   if Inkey$()="y" then goto MainSection
   if Inkey$()="n"
      cls : end
   endif
   REM REFRESH SCREEN
   sync
loop

end
REM *** STOP END SECTION
REM ***************************************
```

Debugging

When programmers find a problem or fault in their program, they generally refer to it as a "bug". Programmers have many different tricks they use to help find where the bug is in the program. One of the most common tricks is one using the print statement. A cleverly placed print command can tell the programmer what part of the code is executing.

VIDEO: Print Statement Trick and Debugging (4:06)

First Debugging

Now watch the following video and take some notes, because you are about to fix a program that has a mistake in it.

VIDEO: Problem with the Program (4:55)

WORKSHEET

A *set cursor 0,50* would have the text placed A: On the left edge of the screen and 50 pixels down from the top.

Your Action:

Load in ***certification1_5.dba*** and fix the repeating print problem in the end Do Loop.

certification1_5.dba

```
REM ****************************************
RemStart
    ***
    *** TITLE - Debugging Program Loops
    *** VERSION - 1.4 LAST UPDATED - 1.1.2000
    *** DEVELOPER - Jason Holm
    *** COPYRIGHT - Ingenious Student Labs
    *** DATE CREATED - 1.1.2000
    ***
RemEnd
REM *** START SYSTEM SETUP SECTION

sync on : sync rate 30
autocam on
hide mouse

REM *** STOP SYSTEM SETUP SECTION
REM ****************************************

REM ****************************************
REM *** START INTRO SECTION
REM *** INTRO SECTION HEADER

REM SET VARIABLES
REM SCREEN DISPLAY
cls
load bitmap "images/cubetitle.bmp"
print "Copyright (c) Ingenious Student Labs"
REM LOAD SOUNDS
REM SPECIAL EFFECTS
REM REFRESH SCREEN
sync

REM *** INTRO SECTION LOOP

do
   REM SCREEN DISPLAY
   set cursor 0,40 ⬅ This line resets the position where the text will print
      RemStart
      What happens if we REM the SET CURSOR command
      in a loop?
      RemEnd
   print "The Introduction Loop"
   REM CONTROL INPUT
   x=scancode()
   if Keystate(x)=1 then goto MainSection
   REM REFRESH SCREEN
   sync
loop
```

```
REM *** END INTRO SECTION
REM ***************************************

REM ***************************************
REM *** START MAIN SECTION
REM *** MAIN SECTION HEADER

MainSection:
REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls 0 : backdrop on
REM LOAD TEXTURES
REM LOAD SOUNDS
REM OBJECT CREATION
make object cube 1,100
REM LOAD MODELS
REM SET CAMERA
REM REFRESH SCREEN
sync

REM *** MAIN SECTION LOOP

do
   REM SPECIAL EFFECTS
   REM OBJECT ORIENTATIONS
   1aY# = object angle Y(1)
   REM CAMERA ORIENTATIONS
   REM LIVE SCREEN DISPLAY
   ink rgb(255,255,255),0 : set text font "times" : set text size 16
: set text to bold : set text transparent
   center text 320,440,"USE ARROW KEYS TO MOVE   |   PRESS 'Q' TO
QUIT"
   set cursor 0,0
   print "The Main Loop runs till the Q key is pressed"
   REM CONTROL INPUT
   if Upkey()=1 then move object 1,10
   if Leftkey()=1 then 1aY# = wrapvalue(1aY#-5)
   if Rightkey()=1 then 1aY# = wrapvalue(1aY#+5)
   if Inkey$()="q"
      delete object 1 : backdrop off : goto EndSection
   endif
   REM TRANSFORM OBJECTS
   REM MOVE OBJECTS
   Yrotate object 1,1aY#
   REM CHECK FOR COLLISION
   REM REFRESH SCREEN
   sync
loop

REM *** STOP MAIN SECTION
REM ***************************************
```

```
REM ****************************************
REM *** START END SECTION
REM *** END SECTION HEADER

EndSection:
REM SCREEN DISPLAY
cls 0
ink rgb(255,255,255),0 : set text font "times" : set text size 20 :
set text to bold : set text transparent
center text 320,240,"PLAY AGAIN [Y/N]?"
REM LOAD SOUNDS
REM REFRESH SCREEN
sync

REM *** END SECTION LOOP

do
   REM SCREEN DISPLAY
```

⬅ Enter **set cursor 0,40** here to fix the looping bug

```
   print "End Loop waiting for the Y or N key"
   REM CONTROL INPUT
   if Inkey$()="y" then goto MainSection
   if Inkey$()="n"
      cls : end
   endif
   REM REFRESH SCREEN
   sync
loop

end
REM *** STOP END SECTION
REM ****************************************
```

Certification Module II

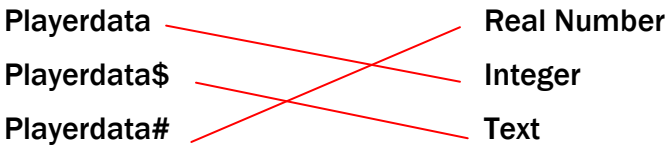
Variables, Math, For/Next Loops, If/Then Statements

Certification II Worksheet

Teacher's Version

Variables

Match the variable with the type of data it can hold.



(found in Variables video)

Math

Using the CLI, have the computer print the answers for the following equations.

2+3*4-1	13	1*2*3*4	24
(2+3)*4-1	19	(1*2)*3*4	24
(2+3)*(4-1)	15	1*(2*3)*4	24
2-10/2+6	-9	(1*2)*(3*4)	24
2-10/(2+6)	1	3/5	0
(2-10)/2+6	2	5/8	0
(2-10)/(2+6)	-1		

Random Numbers are important for programmers because they can be used to help characters in a program to act B .

(found in certification text)

A. The same each time they are encountered
B. Unpredictably
C. Only once and then stop

Certification II Worksheet

Teacher's Version

Using the CLI, fill in the numbers generated by the RND( ) command below for an upper limits of 100 and 20.

Using RND(100)	Using RND(20)
_____	_____
_____	_____
_____	_____
_____	_____
_____	_____

(these will be random)

Looking at the numbers above, do you see any trends or repeat numbers?

Probably not – to see any trend, you must have thousands of numbers generated and a lot of time to look for the trend!

What is the "word" you can use to remember the mathematical operator precedence?

PEMDAS (found in certification text)

List the Mathematical Operators in order of precedence:

( )
^
*
/
+
-

(found in certification text)

Certification II Worksheet

Teacher's Version

List 3 examples of a Real Number:

_____ _____ _____

(anything with a decimal)

List 3 examples of an integer:

_____ _____ _____

(any whole number)

What is the command for creating a random number between 0 and 640?

rnd(640)

(found in certification text and Using Randomly Generated Numbers video)

The command for generating your name onto the screen randomly is:

text rnd(640),rnd(480) ,"your name"

(found in certification text and Using Randomly Generated Numbers video)

For what reason should you use a FOR/NEXT Loop?

A FOR/NEXT Loop should be used when you want to do some action or a series of actions an exact number of times. (found in certification text)

Write a FOR/NEXT loop that will print "hello" onto the screen 12 times.

```
for x = 1 to 12
print "hello"
next x
```

(found in certification text)

Certification II Worksheet

Teacher's Version

Write an IF/THEN that will go to the MainSection if the variable Score is equal to 100 or prints "Waiting" in the center of the screen if Score is any other value.

```
if Score = 100 then goto MainSection
  center text 320,240 ,"Waiting"
endif
```

(found in certification text)

What are the Logical Operators for the following?

=	Equal to
<>	Not Equal to
<	Less than
>	Greater than
<=	Less than or Equal to
>=	Greater than or Equal to

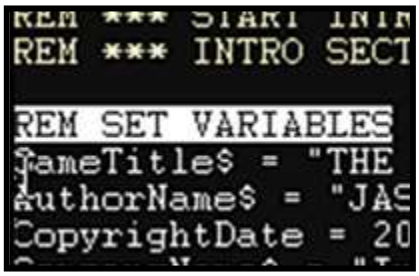
(found in certification text)

In the following IF/THEN statement, the bolded part is called the condition .

(found in certification text)

```
If x > 3 then goto End
Y=y+5
endif
```

Variables



Variables are symbols into which data can be stored. They are called *variables* because the data they represent can be varied and changed. Variables offer programmers the advantage of have a place to store information that is unknown at the time of creating the program.

Think of a variable as a place to store some bit of unknown information. Say you are writing a game program. In your game, you have the player sign in with their name and at the end of the game you want to display their name and score for them. But at the time of writing the game, you don't know who will be playing; you don't know what their name is, let alone what score they might get. So, you use a variable called Player$ to hold the name they type in (text information) when they start and a variable called Playerscore to hold the points they earn (number value) while playing your game (by slewing dragons, grinding a lip, or having the fastest lap). At the end of the game, you can print to the center of the screen their name and score by simply printing the variables you stored the info in.

Center Text 320,240 ,Player$+" "+Playerscore
(this will print the player's name with 2 spaces and then their score)

There are a couple of things to remember when creating variables:

1. **There are restricted words and characters that cannot be used as variables.** Print, Set, Text, Hide, >, =, 1,20 are some examples of restricted words and characters you cannot use as the variable name. Why? Because these words, numbers and characters already mean something else in the programming language.

2. **Be specific with your variable names.** A variable name B1 is very vague, but a variable name like Player$, or Playerscore makes it obvious what information is being held by the variable.

ANIMATION: Chocolate Milk Variables

VIDEO: Variables (6:07)

WORKSHEET

Match the variable with the type of data it can hold.

Playerdata → Integer
Playerdata$ → Text
Playerdata# → Real Number

Your Action:

Load in ***certification2_1.dba***. Change the value of AuthorName$ to your own name and run the program to see the effect of your change.

certification2_1.dba

```
REM ****************************************
RemStart
    ***
    *** TITLE - Variables
    *** VERSION - 2.1 LAST UPDATED - 1.1.2000
    *** DEVELOPER - Jason Holm
    *** COPYRIGHT - Ingenious Student Labs
    *** DATE CREATED - 1.1.2000
    ***
RemEnd
REM *** START SYSTEM SETUP SECTION

sync on : sync rate 30
autocam on
hide mouse

REM *** STOP SYSTEM SETUP SECTION
REM ****************************************

REM ****************************************
REM *** START INTRO SECTION
REM *** INTRO SECTION HEADER

REM SET VARIABLES
GameTitle$ = "THE CUBE GAME" : Rem Set the game title as STRING
VARIABLE GameTitle$
```

⬇ Change the value of this variable to see its effect in the game

```
AuthorName$ = "JASON HOLM" : Rem Set the game author's name as STRING
VARIABLE AuthorName$

CopyrightDate = 2000 : Rem Set the copyright date as INTEGER VARIABLE
CopyrightDate
CompanyName$ = "Ingenious Student Labs" : Rem Set the company name as
STRING VARIABLE CompanyName$
DIM GameCredit$(2) : Rem Define the STRING ARRAY GameCredit$() with
TWO openings
GameCredit$(1) = GameTitle$ : Rem Put GameTitle$ into GameCredit$
opening (1)
GameCredit$(2) = AuthorName$ : Rem Put AuthorName$ into GameCredit$
opening (2)
REM SCREEN DISPLAY
cls 0
ink rgb(255,255,255),rgb(0,150,0) : set text font "arial" : set text
size 36 : set text to bold : set text opaque
center text 320,240," "+GameTitle$+" "
   RemStart
   Alternately, you could use
      center text 320,240," "+GameCredit$(1)+" "
   RemEnd
```

```
ink rgb(255,255,255),0 : set text font "times" : set text size 12 :
set text to italic : set text transparent
text 320,276,"DEVELOPED BY "+AuthorName$ ⬅ Here is where the variable will be used
   RemStart
   Alternately, you could use
      center text 320,276," "+GameCredit$(2)+" "
   RemEnd
ink rgb(255,255,255),0 : set text font "times" : set text size 20 :
set text to bold : set text transparent
center text 320,300,"HIT ANY KEY TO BEGIN"
ink rgb(255,255,255),0 : set text font "arial" : set text size 14 :
set text to normal : set text transparent
center text 320,460,"Copyright (c) "+str$(CopyrightDate)+"
"+CompanyName$
   RemStart
   This demonstrates how to include different types of variables in a
text statement
   RemEnd
REM LOAD SOUNDS
REM SPECIAL EFFECTS
REM REFRESH SCREEN
sync

REM *** INTRO SECTION LOOP

do
   REM SCREEN DISPLAY
   REM CONTROL INPUT
   x=scancode() : if Keystate(x)=1 then goto MainSection
   REM REFRESH SCREEN
   sync
loop

REM *** END INTRO SECTION
REM ****************************************

REM ****************************************
REM *** START MAIN SECTION
REM *** MAIN SECTION HEADER

MainSection:
REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls 0 : backdrop on
REM LOAD TEXTURES
REM LOAD SOUNDS
REM OBJECT CREATION
make object cube 1,100
REM LOAD MODELS
REM SET CAMERA
REM REFRESH SCREEN
sync
```

```
REM *** MAIN SECTION LOOP
do
   REM SPECIAL EFFECTS
   REM OBJECT ORIENTATIONS
   1pX# = object position X(1) : Rem Store the X position of Object 1
as a REAL NUMBER VARIABLE
   1pZ# = object position Z(1) : Rem Store the Z position of Object 1
as a REAL NUMBER VARIABLE
   1aY = object angle Y(1) : Rem Store the Y angle of Object 1 as an
INTEGER VARIABLE
   REM CAMERA ORIENTATIONS
   REM LIVE SCREEN DISPLAY
   ink rgb(255,255,255),0 : set text font "times" : set text size 16
: set text to bold : set text transparent
   center text 320,440,"USE ARROW KEYS TO MOVE   |   PRESS 'Q' TO
QUIT"
   set cursor 0,0 : print "X position: " ; 1pX# ; " | " ; object
position X(1) : Rem Print the X position of Object 1
   set cursor 0,20 : print "Z position: " ; 1pZ# ; " | " ; object
position Z(1) : Rem Print the Z position of Object 1
   set cursor 0,60 : print "Y angle: " ; 1aY ; " | " ; object angle
Y(1) : Rem Print the Y angle of Object 1
   REM CONTROL INPUT
   if Upkey()=1 then move object 1,10
   if Leftkey()=1 then 1aY = wrapvalue(1aY-5)
   if Rightkey()=1 then 1aY = wrapvalue(1aY+5)
   if Inkey$()="q"
      delete object 1 : backdrop off : goto EndSection
   endif
   REM TRANSFORM OBJECTS
   REM MOVE OBJECTS
   Yrotate object 1,1aY
   REM CHECK FOR COLLISION
   REM REFRESH SCREEN
   sync
loop

REM *** STOP MAIN SECTION
REM ************************************
```

```
REM ****************************************
REM *** START END SECTION
REM *** END SECTION HEADER

EndSection:
REM SCREEN DISPLAY
cls 0
ink rgb(255,255,255),0 : set text font "times" : set text size 20 :
set text to bold : set text transparent
center text 320,240,"PLAY AGAIN [Y/N]?"
REM LOAD SOUNDS
REM REFRESH SCREEN
sync

REM *** END SECTION LOOP

do
   REM CONTROL INPUT
   if Inkey$()="y" then goto MainSection
   if Inkey$()="n"
      cls : end
   endif
   REM REFRESH SCREEN
   sync
loop

end
REM *** STOP END SECTION
REM ****************************************
```

Math

Operands or Operators are the characters the programmer uses to tell the computer what type of mathematical operation they want done.

The order in which you do the math matters. Computers follow some simple rules to keep the order straight for a calculation. **PEMDAS** is a word you can remember -- it stands for:

Parentheses
Exponents
Multiplication
Division
Addition
Subtraction

-- which is the order the computer will do the parts of an equation. If you don't think order matters, look at the following equations and you can see the difference.

A. (2*3)-4 = 2
B. 2*3-4 = 2
C. 2*(3-4) = -2

Computation order is important!

All of these equations are correct. It is just the difference in the order in which the calculation is done that matters. A and B are the same because multiplication comes before subtraction (PE**M**DA**S**), but notice the difference in C. Since the parentheses are calculated before multiplication (**P**E**M**DAS), the answer is dramatically different.

WORKSHEET
What is the "word" you can use to remember the mathematical operator precedence? *PEMDAS*

In the equation X=1+2*3, for example, 2 will be multiplied by 3 first. After the calculation of 2*3, the result, 6, is then added to 1; the variable X would be set to 7. If we modify our equation to X=(1+2)*3, the operation inside the parentheses will be calculated first. The result, 3, will then be multiplied by 3 to give a final value for X of 9.

WORKSHEET
List the Mathematical Operators in order of precedence:
( ) , ^ , * , / , + , -

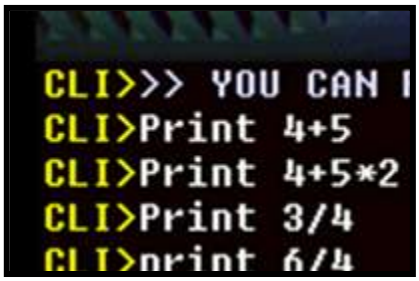
Table of Mathematical Precedence

Operator	Description	Precedence in Calculations
( )	Parenthesis	1
^	Exponentiation	2
*	Multiplication	3
/	Division	4
+	Addition	5
-	Subtraction	6

VIDEO: Equations in Code (2:50)

Your Action:

Use the CLI in Dark Basic to calculate the answers for the list of equations on the worksheet.



Example – into the CLI type:

Print 6/3-2

Hit enter and the answer will appear at the top of the screen. Repeat this process for the equations on the Certification II Worksheet.

WORKSHEET

Match the variable with the type of data it can hold.

*2+3*4-1*	=	*13*	*1*2*3*4*	=	*24*
*(2+3)*4-1*	=	*19*	*(1*2)*3*4*	=	*24*
(2+3)(4-1)*	=	*15*	*1*(2*3)*4*	=	*24*
2-10/2+6	=	*-9*	*(1*2)*(3*4)*	=	*24*
2-10/(2+6)	=	*1*	*3/5*	=	*0*
(2-10)/2+6	=	*2*	*5/8*	=	*0*
(2-10)/(2+6)	=	*-1*			

Note: all the answers are coming back as integers and not real numbers.

Real numbers are all numbers that can be expressed as decimals. Real numbers correspond to every point on the number line and include all rational and irrational numbers. *Integers* are whole numbers, positive or negative, including zero.

Why do programs sometimes use integers and sometimes real numbers? Basically, programmers will use integers (whole numbers) for For/Next loops and use real numbers (floating point) for precise calculations. Also, speed can be an issue; it takes a computer three times longer to calculate with real numbers than with integers.

WORKSHEET

List 3 examples of a Real Number: *1.5, 0.7, 29.073 (anything with a decimal)*

List 3 examples of an integer: *2, 17, 483 (any whole number)*

Random Numbers

Good game programmers try to give the “characters” or bad guys in their games some unpredictability. Say the monsters in a game always turned left when you shot at them, well that would just make it too easy to win, and you would probably consider the monsters too stupid. Quickly you would become bored with the game and probably would not buy another game from that company. One of the uses of random numbers is to give characters a little unpredictability with some artificial intelligence. You can generate a random number, and depending on whether it is odd or even, your monster will turn left or right when dodging a round from a Quantum Plasma Six Shooter.

WORKSHEET
Random Numbers are important for programmers because they can be used to help characters in a program to act *B: unpredictably*.

You would think creating random numbers for a computer would be easy, but it is actually one of the hardest things to do. Computer scientist are trying all the time to get a computer to create truly random numbers, but computers can get close to truly random. But for some basic gaming, the random numbers the computer generates are close enough.

VIDEO: Generating Random Numbers (1:55)

WORKSHEET
What is the command for creating a random number between 0 and 640? rnd(640)

Using the CLI, fill in the numbers generated by the RND( ) command below for an upper limits of 100 and 20. *(these will be random)*

Using **RND(100)** : *3, 24, 75, etc.*
Using **RND(20)** : *5, 12, 19, etc.*

Looking at the numbers above, do you see any trends or repeat numbers?
Probably not -- to see any trend, you must have thousands of numbers generated and a lot of time to look for the trend!

To see how you can replace a value with a random number generator, watch the following video:

VIDEO: Using Randomly Generated Numbers (2:36)

WORKSHEET
The command for generating your name onto the screen randomly is:
text rnd(640),rnd(480) ,"your name"

Your Action:
Load in ***certification1_2.dba***. Edit the DO LOOP in the Introduction section to print your name randomly around the screen using a text command.

certification1_2.dba

```
...
REM *** INTRO SECTION LOOP

do
```

⬅ Insert **text rnd(640),rnd(480) ,"your name"** here to see the effect

```
   REM CONTROL INPUT
   x=scancode() : if Keystate(x)=1 then goto MainSection
   REM REFRESH SCREEN
   sync
loop
...
```

VIDEO: Variables and Random Numbers (3:22)

Your Action:
Load in and run ***certification 2_2.dba***. You will see the random numbers appear on the screen.

certification2_2.dba

```
REM ****************************************
RemStart
    ***
    *** TITLE - Math and Random Rumbers
    *** VERSION - 2.2 LAST UPDATED - 1.1.2000
    *** DEVELOPER - Jason Holm
    *** COPYRIGHT - Ingenious Student Labs
    *** DATE CREATED - 1.1.2000
    ***
RemEnd
REM *** START SYSTEM SETUP SECTION

sync on : sync rate 30
autocam on
hide mouse
randomize timer() : Rem Generate different random numbers each time

REM *** STOP SYSTEM SETUP SECTION
REM ****************************************

REM ****************************************
REM *** START INTRO SECTION
REM *** INTRO SECTION HEADER

REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls
load bitmap "images/cubetitle.bmp"
ink rgb(255,255,255),0 : set text font "arial" : set text size 14 :
set text to normal : set text transparent
center text 320,460,"Copyright (c) 2000 Ingenious Student Labs"
REM LOAD SOUNDS
REM SPECIAL EFFECTS
REM REFRESH SCREEN
sync

REM *** INTRO SECTION LOOP

do
   REM SCREEN DISPLAY
   ⬇ Here is where the random numbers are generated
   randomX = rnd(640) : Rem Set a random variable based on a screen
width
   randomY = rnd(480) : Rem Set a random variable based on a screen
height
   ink rgb(255,0,0),0 : set text font "times" : set text size 12 :
set text to bold : set text transparent
   text randomX,randomY,"X: "+str$(randomX)+" , Y: "+str$(randomY)
   ⬆ Here is where the random numbers are used to print to the screen
```

```
      Rem Print a phrase in a random place on the screen, and its
coordinates

   REM CONTROL INPUT
   x=scancode() : if Keystate(x)=1 then goto MainSection
   REM REFRESH SCREEN
   sync
loop

REM *** END INTRO SECTION
REM ***************************************

REM ***************************************
REM *** START MAIN SECTION
REM *** MAIN SECTION HEADER

MainSection:
REM DECLARE VARIABLES
MyDistance = 0
REM SCREEN DISPLAY
cls 0 : backdrop on
REM LOAD TEXTURES
REM LOAD SOUNDS
REM OBJECT CREATION
make object cube 1,100
REM LOAD MODELS
REM SET CAMERA
REM REFRESH SCREEN
sync

REM *** MAIN SECTION LOOP

do
   REM SPECIAL EFFECTS
   REM OBJECT ORIENTATIONS
   1pX# = object position X(1)
   1pZ# = object position Z(1)
   1aY = object angle Y(1)
   ClockHeading = 1aY/30 : Rem Divide the Y angle of Object 1 by 30
and store as an INTEGER VARIABLE (1 to 12)
   If ClockHeading = 0 then ClockHeading = 12 : Rem There is no 0
o'clock, so adjust accordingly
   REM CAMERA ORIENTATIONS
```

```
   REM LIVE SCREEN DISPLAY
   ink rgb(255,255,255),0 : set text font "times" : set text size 16
: set text to bold : set text transparent
   center text 320,440,"USE ARROW KEYS TO MOVE   |   PRESS 'Q' TO
QUIT"
   set cursor 0,0 : print "X position: " ; 1pX#
   set cursor 0,20 : print "Z position: "  ;1pZ#
   set cursor 0,60 : print "Y angle: " ; 1aY
   set cursor 0,80 : print "Clock Heading: " ; ClockHeading ; "
o'clock" : Rem Print the clock heading
   set cursor 0,100 : print "I have traveled " ; MyDistance ; "
spaces so far" : Rem Print how far you've moved
   REM CONTROL INPUT
   if Upkey()=1
      move object 1,10
      MyDistance = MyDistance + 10
   endif
   if Leftkey()=1 then 1aY = wrapvalue(1aY-5)
      RemStart
      The function 'wrapvalue' will convert a variable to an angle
from 0 to 360
      For example: wrapvalue(320+90) would equal 50
      (a 410 degree angle would still point 50 degrees from start)
      RemEnd
   if Rightkey()=1 then 1aY = wrapvalue(1aY+5)
   if Inkey$()="q"
      delete object 1 : backdrop off : goto EndSection
   endif
   REM TRANSFORM OBJECTS
   REM MOVE OBJECTS
   Yrotate object 1,1aY
   REM CHECK FOR COLLISION
   REM REFRESH SCREEN
   sync
loop

REM *** STOP MAIN SECTION
REM ************************************
```

```
REM ****************************************
REM *** START END SECTION
REM *** END SECTION HEADER

EndSection:
REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls 0
ink rgb(255,255,255),0 : set text font "times" : set text size 20 :
set text to bold : set text transparent
center text 320,240,"PLAY AGAIN [Y/N]?"
REM LOAD SOUNDS
REM SPECIAL EFFECTS
REM REFRESH SCREEN
sync

REM *** END SECTION LOOP

do
   REM CONTROL INPUT
   if Inkey$()="y" then goto MainSection
   if Inkey$()="n"
      cls : end
   endif
   REM REFRESH SCREEN
   sync
loop

end
REM *** STOP END SECTION
REM ****************************************
```

FOR/NEXT Loops

If you want the program to do an action for a certain number of times, the For/Next Loop is the perfect tool. An example of a FOR/NEXT loop is a car race.

ANIMATION: Race Car Laps

The race lasts for only a certain number of laps (loops) and then the race is over.

The FOR/NEXT loop can be told how many laps it will take, and on each lap it will do the list of tasks set for it. Take a look at the following example:

For x = 1 to 10
List of tasks or actions
Next x

This loop will execute 10 times and then stop. The computer looks at it this way:

X=1
Next X
X=2
Next X
X=3
Next X
X=4….

When X gets to 10, the For/Next Loop condition of X=1 to 10 is satisfied, the loop simply stops, and the program moves to the next command below the loop.

WORKSHEET
For what reason should you use a FOR/NEXT Loop?
A FOR/NEXT Loop should be used when you want to do some action or a series of actions an exact number of times.

VIDEO: Inserting a FOR/NEXT Loop (2:21)

Your Action:
Load in ***certification 1_2.dba*** and create your own FOR/NEXT loop.

Right under the REM *** INTRO SECTION LOOP, type the following:

```
For x=1 to 10
  print x
next x
```

Now run the program and you will see the loop will print the numbers 1 through 10 for you. Now, go back and change the loop so it will run 15 times.

certification1_2.dba

```
...
REM *** INTRO SECTION LOOP
```

Insert
For x=1 to 10
print x
next x
here to see the effect

```
do
   REM CONTROL INPUT
   x=scancode() : if Keystate(x)=1 then goto MainSection
   REM REFRESH SCREEN
   sync
loop
...
```

WORKSHEET
Write a FOR/NEXT loop that will print "hello" onto the screen 12 times.

```
for x = 1 to 12
   print "hello"
next x
```

Let's go look at how a FOR/NEXT Loop is used to make an object grow and shrink while a game is running.

Your Action:
Load in and run ***certification2_3.dba***. Notice how the Cube is pulsing – that is being done by a FOR/NEXT Loop.

To see how that is done, watch this video:

VIDEO: FOR/NEXT Loops in a Game (3:20)

certification2_3.dba

```
REM ****************************************
RemStart
    ***
    *** TITLE - For/Next Loop
    *** VERSION - 2.3 LAST UPDATED - 1.1.2000
    *** DEVELOPER - Jason Holm
    *** COPYRIGHT - Ingenious Student Labs
    *** DATE CREATED - 1.1.2000
    ***
RemEnd
REM *** START SYSTEM SETUP SECTION

sync on : sync rate 30
autocam on
hide mouse
randomize timer()

REM *** STOP SYSTEM SETUP SECTION
REM ****************************************

REM ****************************************
REM *** START INTRO SECTION
REM *** INTRO SECTION HEADER

REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls
load bitmap "images/cubetitle.bmp"
ink rgb(255,255,255),0 : set text font "arial" : set text size 14 :
set text to normal : set text transparent
center text 320,460,"Copyright (c) 2000 Ingenious Student Labs"
sync
REM LOAD SOUNDS
REM SPECIAL EFFECTS
for x = 1 to 10 : Rem Do the following action ten times
   randomX = rnd(640)
   randomY = rnd(480)
   ink rgb(255,0,0),0 : set text font "times" : set text size 12 :
set text to bold : set text transparent
   text randomX,randomY,"X: "+str$(randomX)+" , Y: "+str$(randomY)
next x : Rem Each time you get here, go back and do the action again
until x = 10
REM REFRESH SCREEN
sync
```

```
REM *** INTRO SECTION LOOP

do
   REM SCREEN DISPLAY
   REM CONTROL INPUT
   x=scancode() : if Keystate(x)=1 then goto MainSection
   REM REFRESH SCREEN
   sync
loop

REM *** END INTRO SECTION
REM ************************************

REM ************************************
REM *** START MAIN SECTION
REM *** MAIN SECTION HEADER

MainSection:
REM DECLARE VARIABLES
MyDistance = 0
REM SCREEN DISPLAY
cls : backdrop on
REM LOAD TEXTURES
REM LOAD SOUNDS
REM OBJECT CREATION
make object cube 1,100
REM LOAD MODELS
REM SET CAMERA
REM REFRESH SCREEN
sync

REM *** MAIN SECTION LOOP

do
   REM SPECIAL EFFECTS
   for pulse = 0 to 5 : Rem Do the following action five times
```

⬆ The variable **pulse** is defined by this **for** statement

```
      REM OBJECT ORIENTATIONS
      1pX# = object position X(1)
      1pZ# = object position Z(1)
      1aY = object angle Y(1)
      REM CAMERA ORIENTATIONS
      REM LIVE SCREEN DISPLAY
      ink rgb(255,255,255),0 : set text font "times" : set text size
16 : set text to bold : set text transparent
      center text 320,440,"USE ARROW KEYS TO MOVE   |   PRESS 'Q' TO
QUIT"
      set cursor 0,0 : print "X position: " ; 1pX#
      set cursor 0,20 : print "Z position: " ; 1pZ#
      set cursor 0,60 : print "Y angle: " ; 1aY
```

```
      REM CONTROL INPUT
      if Upkey()=1 then move object 1,10
      if Leftkey()=1 then 1aY = wrapvalue(1aY-5)
      if Rightkey()=1 then 1aY = wrapvalue(1aY+5)
      if Inkey$()="q"
         delete object 1 : backdrop off : goto EndSection
      endif
      REM TRANSFORM OBJECTS
      Rem Make Object 1 pulse by growing and contracting
      size = (pulse*2)+100
```

⬆ The variable **size** changes based on the variable **pulse**

```
         RemStart
         The variable 'size' takes the variable 'pulse'
         (which increases by 1 each time it gets here, and resets
every five times)
         multiplies that number by two (giving us 2,4,6,8, or 10)
         and adding 100 to that number (giving us 102,104,106,108, or
110)
         RemEnd
      scale object 1,size,size,size
```

⬆ The cube changes size based on the variable **size**

```
         RemStart
         This function takes the variable 'size' and scales object 1
to fit that size,
         Making the cube a little bigger each time.
         After five times, the cube shrinks back to the first size
(102)
         and repeats the process again.
         This makes the cube appear to pulse
         RemEnd
      REM MOVE OBJECTS
      Yrotate object 1,1aY
      REM CHECK FOR COLLISION
      REM REFRESH SCREEN
      sync
```

⬇ The variable **pulse** is incremented by this **next** statement

```
   next pulse : Rem Each time you get here, go back and do the action
again until x = 5
loop

REM *** STOP MAIN SECTION
REM ************************************
```

```
REM ****************************************
REM *** START END SECTION
REM *** END SECTION HEADER

EndSection:
REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls 0
ink rgb(255,255,255),0 : set text font "times" : set text size 20 :
set text to bold : set text transparent
center text 320,240,"PLAY AGAIN [Y/N]?"
REM LOAD SOUNDS
REM SPECIAL EFFECTS
REM REFRESH SCREEN
sync

REM *** END SECTION LOOP

do
   REM CONTROL INPUT
   if Inkey$()="y" then goto MainSection
   if Inkey$()="n"
      cls : end
   endif
   REM REFRESH SCREEN
   sync
loop

end
REM *** STOP END SECTION
REM ****************************************
```

IF/THEN Statements

For a program to be able to be a game, it has to make many decisions - quickly - based on internal data, external data (online games) or inputs from the game players. One of the most popular decision tools programmers have is the IF/THEN statement. An IF/THEN is easily described as "IF this is true THEN do this". The IF part is set up as a condition. Something like IF X = 2, or IF X > 0, or IF NAME$ = "TOM". For the first example, if X was indeed equal to 2 then the condition is said to be true. The IF statement would then move on to the THEN part. But if the IF statement was false, the program skips the THEN part and continues to the next line of code.

WORKSHEET

In the following IF/THEN statement, the bolded part is called the *condition*.

if **x > 3** then goto End

There are two ways you can use an IF/THEN statement:

First, the IF/THEN can be used as a single line of code to make a simple decision.

```
IF X = 3 THEN print x
```

or

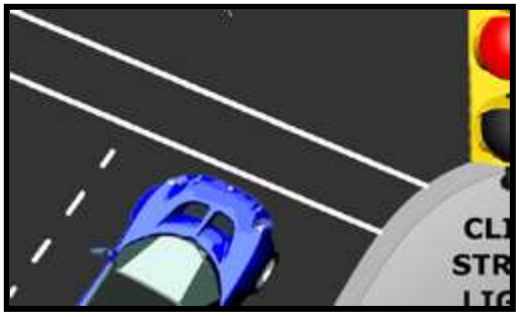
```
IF X = 3 THEN print x : print y : z = x + y
```

Second, the IF/THEN can be used to make a decision and then run multiple lines of code.

```
IF X = 3
   print x
   print y
   z = x + y
ENDIF
```

If the condition (x=3) is TRUE, then all the actions between the IF statement and the ENDIF statement will be executed. The statement THEN is not written, but all the code between the IF and the ENDIF is still referred to the THEN part.

ANIMATION: Stoplight



In the animation of the stoplight, the car was behaving using the following IF/THEN:

```
IF Light = Green
   Drive Forward
ENDIF

IF Light = Red
   Stop Driving
ENDIF
```

IF/THEN is a command that looks at a condition (example Light = Green). If the condition is TRUE, then the IF goes to the THEN part. If the condition is FALSE, the IF skips the THEN and ENDIF parts, and continues to the next line of code below it.

However, there is another way of making an either/or condition:

```
IF Light = Green
    Drive Forward
ELSE
    Stop Driving
ENDIF
```

If the condition is TRUE, then the IF goes to and runs the THEN part (in this case, the Drive Forward action), stopping if it reaches an ELSE or an ENDIF statement.

If the condition is FALSE, the IF skips the THEN part and searches for an ELSE statement. If it finds an ELSE statement, it runs the code between the ELSE and the ENDIF (in this case, the Stop Driving action).

If the condition is FALSE, but there is no ELSE statement, the IF simply jumps to the ENDIF and continues with the program.

Therefore, the IF/THEN can be used as a branched decision:

```
IF score > 1000
    goto Level2
ELSE
    Y = Inkey()
    Score = Y + X
ENDIF
```

 *WORKSHEET*

Write an IF/THEN that will go to the MainSection if the variable Score is equal to 100 or prints "Waiting" in the center of the screen if Score is any other value.

```
if Score = 100
    goto MainSection
else
    center text 320,240 ,“Waiting”
endif
```

Logical Operators

IF/THENs look at the logic of the condition. Just like there are Mathematical Operators you use with equations, there are Logical Operators you use with the IF/THEN.

Logical Operators	
=	Equal to
<>	Not Equal to
<	Less Than
>	Greater Than
<=	Less Than or Equal to
>=	Greater Than or Equal to

WORKSHEET

What are the Logical Operators for the following?

=	*Equal to*
<>	*Not Equal to*
<	*Less than*
>	*Greater than*
<=	*Less than or Equal to*
>=	*Greater than or Equal to*

The Logical Operators are what defines the relationship inside a condition. For Example:

X = 1	If the value of x is 1, then this condition is TRUE
Larry > 3	If the value of Larry is greater than 3 then this condition is TRUE
Score <= 1000	If the value of Score is less than or equal to 1000 then this condition is TRUE

VIDEO: IF/THEN Loops in a Game (4:29)

Your Action:
Load in ***certification2_4.dba*** and examine where the IF/THEN loops are located.

VIDEO: Inserting IF/THEN Loops (4:13)

Your Action:
Go back into ***certification2_4.dba*** and add an IF/THEN state to stop the game if the Cube gets too far along the X axis.

First thing is to find out the name of the variable that keeps the X axis position information. You will find that in the top part of the Main Loop – See below.

```
REM OBJECT ORIENTATIONS
1pX# = object position X(1)
```

We see that 1pX# is the variable for the x position.

Now that we know this, we can write our IF/THEN statement. In the THEN part of our statement, we need to shut down the game's 3D world. To do that we will delete the object and turn off the background before we go to the end loop. See below.

```
if 1pX# > 200 then delete object 1 : backdrop off : goto EndSection
```

Take the IF/THEN above and insert it into ***certification2_4.dba*** game right before the end of the Main Loop (just above the loop command). Run the game and see what happens when the X value gets to 200. Afterwards you can go back and change 200 in the condition to something larger or smaller.

certification2_4.dba

```
REM ****************************************
RemStart
    ***
    *** TITLE - If/Then Statements
    *** VERSION - 2.4 LAST UPDATED - 1.1.2000
    *** DEVELOPER - Jason Holm
    *** COPYRIGHT - Ingenious Student Labs
    *** DATE CREATED - 1.1.2000
    ***
RemEnd
REM *** START SYSTEM SETUP SECTION

sync on : sync rate 30
autocam on
hide mouse
randomize timer()

REM *** STOP SYSTEM SETUP SECTION
REM ****************************************

REM ****************************************
REM *** START INTRO SECTION
REM *** INTRO SECTION HEADER

REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls
load bitmap "images/cubetitle.bmp"
ink rgb(255,255,255),0 : set text font "arial" : set text size 14 :
set text to normal : set text transparent
center text 320,460,"Copyright (c) 2000 Ingenious Student Labs"
REM LOAD SOUNDS
REM SPECIAL EFFECTS
REM REFRESH SCREEN
sync

REM *** INTRO SECTION LOOP

do
   REM SCREEN DISPLAY
   REM CONTROL INPUT
   x=scancode()
   if Keystate(x)=1 then goto MainSection : Rem If the player hits
any key, then start the game ⬅ IF/THEN Statement
   REM REFRESH SCREEN
   sync
loop

REM *** END INTRO SECTION
REM ****************************************
```

```
REM ****************************************
REM *** START MAIN SECTION
REM *** MAIN SECTION HEADER

MainSection:
REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls : backdrop on
REM LOAD TEXTURES
REM LOAD SOUNDS
REM OBJECT CREATION
make object cube 1,100
REM LOAD MODELS
REM SET CAMERA
REM REFRESH SCREEN
sync

REM *** MAIN SECTION LOOP

do
   REM SPECIAL EFFECTS
   REM OBJECT ORIENTATIONS
   1pX# = object position X(1)  ⬅ Player Cube's X Position
   1pZ# = object position Z(1)
   1aY = object angle Y(1)
   REM CAMERA ORIENTATIONS
   REM LIVE SCREEN DISPLAY
   ink rgb(255,255,255),0 : set text font "times" : set text size 16
: set text to bold : set text transparent
   center text 320,420,"USE ARROW KEYS TO MOVE   |   PRESS SPACE BAR
TO CHANGE COLOR"
   center text 320,440,"PRESS 'Q' TO QUIT"
   set cursor 0,0 : print "X position: " ; 1pX#
   set cursor 0,20 : print "Z position: " ; 1pZ#
   set cursor 0,60 : print "Y angle: " ; 1aY
   ⬇ IF Statement
   if 1pZ# > 500 : Rem If Object 1 gets too far away from the viewer
      center text 320,100,"COME BACK! YOU'RE TOO FAR AWAY!" : Rem
Then print a message on screen
   endif : Rem Continue the program  ⬅ ENDIF Statement
```

```
   REM CONTROL INPUT
   ⬇ IF/THEN Statement
   if Upkey()=1 then move object 1,10 : Rem If the player hits the
'UP' arrow key, then move forward 10 units
   ⬇ IF/THEN Statement
   if Downkey()=1 then move object 1,-10 : Rem If the player hits the
'DOWN' arrow key, then move backward 10 units
   ⬇ IF/THEN Statement
   if Leftkey()=1 then 1aY = wrapvalue(1aY-5) : Rem If the player
hits the 'LEFT' arrow key, then turn left 5 units
   ⬇ IF/THEN Statement
   if Rightkey()=1 then 1aY = wrapvalue(1aY+5) : Rem If the player
hits the 'RIGHT' arrow key, then turn right 5 units
   ⬇ IF Statement
   if Spacekey()=1 : Rem If the user hits the space bar
      red=rnd(255) : Rem Set the variable 'red' as a random number
between 0 and 255
      green=rnd(255) : Rem Set the variable 'green' as a random
number between 0 and 255
      blue=rnd(255) : Rem Set the variable 'blue' as a random number
between 0 and 255
      color object 1,rgb(red,green,blue) : Rem Change the cube to a
random color
   endif ⬅ ENDIF Statement
   ⬅ IF/THEN Statement
   if Inkey$()="q" : Rem If the player hits the 'q' key
      delete object 1 : backdrop off : goto EndSection : Rem Then
clear the 3D world and go to the end screen
   endif : Rem Continue the program ⬅ ENDIF Statement
   REM TRANSFORM OBJECTS
   REM MOVE OBJECTS
   Yrotate object 1,1aY
   REM CHECK FOR COLLISION
   REM REFRESH SCREEN
   sync

      ⬅ insert   if 1pX# > 200 then delete object 1 : backdrop off : goto EndSection

loop

REM *** STOP MAIN SECTION
REM ************************************
```

```
REM ****************************************
REM *** START END SECTION
REM *** END SECTION HEADER

EndSection:
REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls 0
ink rgb(255,255,255),0 : set text font "times" : set text size 20 :
set text to bold : set text transparent
center text 320,240,"PLAY AGAIN [Y/N]?"
REM LOAD SOUNDS
REM SPECIAL EFFECTS
REM REFRESH SCREEN
sync

REM *** END SECTION LOOP

do
   REM CONTROL INPUT
   ⬇ IF/THEN Statement
   if Inkey$()="y" then goto MainSection : Rem If the player hits the
'y' key, then start the game again
   ⬇ IF Statement
   if Inkey$()="n" : Rem If the player hits the 'n' key
      cls : end : Rem Then clear the screen and end the program
   endif : Rem Continue the program ⬅ ENDIF Statement
   REM REFRESH SCREEN
   sync
loop

end
REM *** STOP END SECTION
REM ****************************************
```

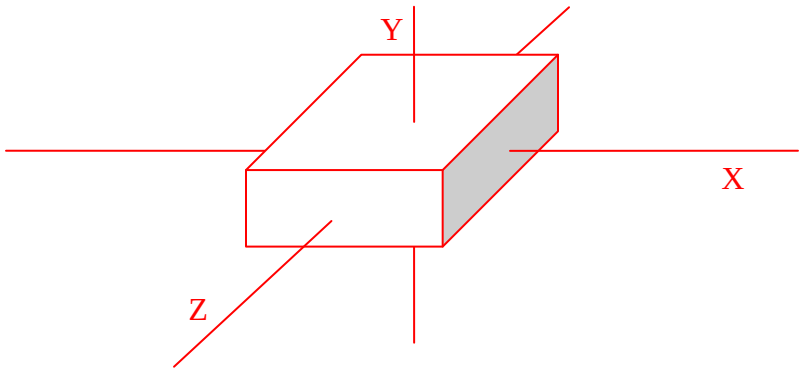
Certification Module III

2D and 3D Game Worlds, Objects, and Cameras

Certification III Worksheet

Teacher's Version

Load in certification3_2.dba and run it. Sketch the object and each axis and carefully title the X,Y,Z axes in the certification3_2.dba game.



Match the X, Y, and Z axes with the cube's surface:

Front and back	Z
Left and right	X
Top and bottom	Y

(found in certification3_1-demo.dba)

What are two advantages of programming a game in 3D?

- You can view from any direction
- More realistic, allowing player to move around in the world

(found in certification text)

What are two advantages of programming a game in 2D?

- Easier and faster for computer to process; better for slower/older computers
- Easier to play; players don't get lost so easily

(found in certification text)

What are three basic characteristics an object can have?

Shape, size, color, position, texture

(found in Objects video)

Certification III Worksheet

Teacher's Version

If you were creating a game about skate boarding, name 5 objects you would want to create to use.

Up to the student, but one object should be the skater!

When you make an object you must give it a unique number so it will not become confused with any other object. (found in Objects video)

What is the command for making an object?

Make object cube 1,100

(found in certification text and Objects video)

What is the command for coloring an object?

Color object #,rgb(0,0,0)

(found in certification text and Objects video)

Use the table below to experiment with position and direction in the 3D world. Using the certification3_2demo.dba file, move the ship to the positions shown below and then describe the position and direction relative to the origin.

Position Table				
Axis Position			Facing Y Angle	Description of the Position and Direction relative to the origin
X	Y	Z		
100	150	300	180	The ship is right, back and above the origin. The ship is facing toward the player.
-100	-45	-100	225	The ship is left, forward, and below the origin. The ship is facing towards the player and slightly to the left.
100	100	-100	90	The ship is right, forward, and above the origin. The ship is facing right.
-150	-225	240	270	The ship is left, back, and below the origin. The ship is facing left.

(found through experimenting with certification3_2demo.dba)

Certification III Worksheet

Teacher's Version

What is the command for loading in a pre-made model to be used in a game?

Load object "walk.x",1

(found in certification text and Animated Objects video)

The following color combinations give you what color?

(0,0,0)	black
(255,255,255)	white

What combination of colors will give you purple?

(255 , 0 , 255)

(found through experimenting with the CLI)

What is the big challenge of creating a walking object?

Making sure the action is smooth. A walk animation is looped. If the loop/animation doesn't begin and end exactly the same way, you will notice the character jerking every step.

(found in Animated Objects video)

Certification III Worksheet

Teacher's Version

List the 5 objects found in certification3_3.dba, along with their object ID number and color.

Object	Object ID Number	Object Color
Cube	1	255,0,0 (Red)
Cylinder	3	255,0,0 (Red)
Cylinder	4	0,255,0 (Green)
Cylinder	5	0,0,255 (Blue)
Cube	6	0,0,255 (Blue)

(found in code of certification3_3.dba in the Main Header section)

FOLLOW ME = 1 is the command to have the camera do what?

Tells the camera to follow object 1

(found in Cameras in a Game video)

If you wanted to position a camera high above and look down at the center of the 3D world, how would you complete the following code?

POSITION CAMERA 0,1500,0

POINT CAMERA 0,0,0

(found in Cameras in a Game video)

2D and 3D Game Worlds

What is the difference between 2D and 3D Game Worlds?

Video game developers use 2-dimensional (2D) and 3-dimensional (3D) ways to display data to the player. Both are viewed on a 2D screen -- the monitor. The difference is how the data of the game (character positions, objects, etc.) is stored. Typically, 2D games store all of the data in pre-made image files. In 3D games, you display 3D images generated from information (3D data) stored about the object (vertices, polygons, textures, scale, etc).

A 3D game will let you see a 3D object from virtually any angle, but this takes quite a bit of mathematical computation. A 2D game can show you an object from different angles, but each "View Angle" requires another pre-drawn 2D image of the object as seen by this angle. Depending on the quality the game designer wants, a 2D game displaying as if it is a 3D game might require 8-18 different 2D images for each object.

So, the big difference between 2D and 3D is how the data is stored. 3D stores 3D data on the points that make up the 3D object. 2D games store a pre-drawn 2D image of the object.

Advantages of 3D

3D is very flexible and allows the game designer to give the player the opportunity to move about and to view the 3D world from every angle. 3D allows the programmer to create animated objects that can be viewed from any angle. It is easy to create new objects in 3D and to store them efficiently by storing the vertices (corners), points and textures that make up the object.

WORKSHEET

What are two advantages of programming a game in 3D?

You can view from any direction

More realistic, allowing player to move around in the world

Advantages of 2D

Currently, 2D does have some advantages over 3D. 2D displays quicker than 3D and therefore 2D games may work on older and slower computers more easily than a 3D game. Some players struggle with playing games in 3D space and become confused or lost in the 3D worlds.

WORKSHEET

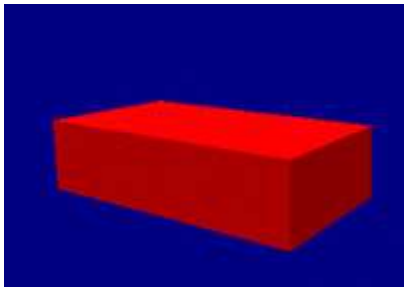
What are two advantages of programming a game in 2D?

Easier and faster for computer to process; better for slower/older computers

Easier to play; players don't get lost so easily

Objects

Objects are any 3D elements in your game. An object takes up space in your 3D game world and it has a number of other attributes. Some basic attributes are:



- Size
- Color
- Texture
- Shape or Model
- Collision Radius

VIDEO: Objects (1:18)

WORKSHEET

What are three basic characteristics an object can have?
Shape, size, color, position, texture

If you were creating a game about skate boarding, name 5 objects you would want to create to use.
Up to the student, but one object should be the skater!

When you make an object you must give it a unique *number* so it will not become confused with any other object.

What is the command for making an object?
make object cube 1,100

What is the command for coloring an object?
color object #,rgb(0,0,0)

3D objects can be shaped by changing their scale along one of their axes. In a 3D world, all points have a position in 3D space determined by 3 axes X,Y,Z (see the connection – 3D world – 3 axes). An object's position is not only determined by X,Y,Z, but the object's size can be scaled differently along each axis.

Your Action:
Load and run ***certification3_1.dba***. Drive the block around. Notice how the position and direction of the box changes. Use the space bar to change the color of the box.

certification3_1.dba

```
REM ****************************************
RemStart
    ***
    *** TITLE - 3D Objects
    *** VERSION - 3.1 LAST UPDATED - 1.1.2000
    *** DEVELOPER - Jason Holm
    *** COPYRIGHT - Ingenious Student Labs
    *** DATE CREATED - 1.1.2000
    ***
RemEnd
REM *** START SYSTEM SETUP SECTION

sync on : sync rate 30
autocam off
hide mouse
randomize timer()

REM *** STOP SYSTEM SETUP SECTION
REM ****************************************

REM ****************************************
REM *** START INTRO SECTION
REM *** INTRO SECTION HEADER

REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls
load bitmap "images/cubetitle.bmp"
ink rgb(255,255,255),0 : set text font "arial" : set text size 14 :
set text to normal : set text transparent
center text 320,460,"Copyright (c) 2000 Ingenious Student Labs"
REM LOAD SOUNDS
REM SPECIAL EFFECTS
REM REFRESH SCREEN
sync

REM *** INTRO SECTION LOOP

do
   REM SCREEN DISPLAY
   REM CONTROL INPUT
   x=scancode() : if Keystate(x)=1 then goto MainSection
   REM REFRESH SCREEN
   sync
loop

REM *** END INTRO SECTION
REM ****************************************
```

```
REM ****************************************
REM *** START MAIN SECTION
REM *** MAIN SECTION HEADER

MainSection:
REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls
backdrop on
REM OBJECT CREATION
make object cube 1,100 : Rem Make Object 1 a cube, 100 units in size
   rem Make object CUBE Object Number, Size Value
   rem Make object BOX Object Number, Width, Height, Depth
   rem Make object CYLINDER Object Number, Size Value
   rem Make object CONE Object Number, Size Value
   rem Make object PLAIN Object Number, Width Value, Height Value
   rem Make object TRIANGLE Object Number, X1, Y1, Z1, X2, Y2, Z2,
X3, Y3, Z3
   color object 1,rgb(255,0,0) : Rem Color Object 1 Red

      Rem Resize Object 1
      scale object 1, 100,50,200
      remStart
      The object is scaled
         100% along the X axis
          50% along the Y axis
         200% along the Z axis
         To find the new size, calculate the percentage of the
original (100 units)
      remEnd

   position object 1,50,0,50 : Rem Put Object 1 at a specific place
in the world
   rotate object 1,0,45,0 : Rem Rotate Object 1 a specific amount
REM LOAD MODELS
REM SET CAMERA
position camera -20,100,-300
point camera 0,50,0
REM REFRESH SCREEN
sync
```

```
REM *** MAIN SECTION LOOP

do
   REM SPECIAL EFFECTS
   REM OBJECT ORIENTATIONS
   1pX# = object position X(1)
   1pZ# = object position Z(1)
   1aY = object angle Y(1)
   REM CAMERA ORIENTATIONS
   REM LIVE SCREEN DISPLAY
   ink rgb(255,255,255),0 : set text font "times" : set text size 16
: set text to bold : set text transparent
   center text 320,420,"USE ARROW KEYS TO MOVE   |   PRESS SPACE BAR
TO CHANGE COLOR"
   center text 320,440,"PRESS 'Q' TO QUIT"
   set cursor 0,0
   print "OBJECT 1:"
   print " X position: " ; 1pX#
   print " Z position: " ; 1pZ#
   print
   print " Y angle: " ; 1aY
   REM CONTROL INPUT
   if Upkey()=1 then move object 1,10
   if Downkey()=1 then move object 1,-10
   if Leftkey()=1 then 1aY = wrapvalue(1aY-5)
   if Rightkey()=1 then 1aY = wrapvalue(1aY+5)
   if Inkey$()="q"
      delete object 1 : Rem Delete Object 1 from the game field
      backdrop off
      goto EndSection
   endif
   red=rnd(255) : green=rnd(255) : blue=rnd(255)
   if Spacekey()=1 then color object 1,rgb(red,green,blue)
   REM TRANSFORM OBJECTS
   REM MOVE OBJECTS
   Yrotate object 1,1aY
   REM CHECK FOR COLLISION
   REM REFRESH SCREEN
   sync
loop

REM *** STOP MAIN SECTION
REM ************************************
```

```
REM ****************************************
REM *** START END SECTION
REM *** END SECTION HEADER

EndSection:
REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls 0
ink rgb(255,255,255),0 : set text font "times" : set text size 20 :
set text to bold : set text transparent
center text 320,240,"PLAY AGAIN [Y/N]?"
REM LOAD SOUNDS
REM SPECIAL EFFECTS
REM REFRESH SCREEN
sync

REM *** END SECTION LOOP

do
   REM CONTROL INPUT
   if Inkey$()="y" then goto MainSection
   if Inkey$()="n"
      cls : end
   endif
   REM REFRESH SCREEN
   sync
loop

end
REM *** STOP END SECTION
REM ****************************************
```

To use an object, you must first create the object in your 3D game world. The command's format is:

MAKE OBJECT TYPE #,SIZE

Each object you make must have its own unique ID number (#). When you work with objects, it is a good idea to make a list somewhere of the object and its number.

Here are some examples of object construction or MAKE commands:

MAKE OBJECT CUBE 1,100
MAKE OBJECT CONE 2,50
MAKE OBJECT CYLINDER 3,75

You may want to use something other than the geometric objects above, like an alien or tank. To do this, you will load in a pre-made model from a library of models that have been created for the game. In the game industry, there are people who do nothing but create the objects that others use when creating games. Here is an example of how to bring in a pre-made model from a library.

LOAD OBJECT "models/walk.x",1

Objects can also have colors. To set the color of an object, you use the COLOR OBJECT command. It's format is:

COLOR OBJECT #,RGB(###,###,###)

RGB stands for **R**ed, **G**reen and **B**lue. It is the combination of these three colors that creates the color of the object. Each color has a range of 0-255. For example, if you want an object to be red, then you could simply use the combination (255,0,0). Blue is (0,0,255).

COLOR OBJECT 1,rgb(120,120,0) will make OBJECT 1 dark yellow in color.

Your Action:
Use the CLI to make a CUBE, CONE and CYLINDER. Between each, use a CLS command to clear the screen. Don't worry that it may seem that you are too close to the object. Later you will learn about cameras and views and what to do about that.

MAKE OBJECT CUBE 1,100
MAKE OBJECT CONE 2,50
MAKE OBJECT CYLINDER 3,75
MAKE OBJECT SPHERE 4,75

Now go and color one of these objects. Make a new object with a new number, and then use the OBJECT COLOR command to change the objects color a few times.

WORKSHEET
The following color combinations give you what color?
(0,0,0) *black*
(255,255,255) *white*

What combination of colors will give you purple?
(255 , 0 , 255)

Objects can also be scaled along their axes.

Your Action:
Load and run ***certification3_1-demo.dba***. Give the object a different size along each axis. As you move the object around with the arrow keys, you will notice that the object has a front. Notice the different axes relate to the object's top and bottom, left and right, and front and back. Change the scale values for each of the axes and see what effect it has.

WORKSHEET
Match the X, Y, and Z axes with the cube's surface:
Front and back: *Z*
Left and right: *X*
Top and bottom: *Y*

certification3_1-demo.dba

```
REM ****************************************
RemStart
    ***
    *** TITLE - Scaling Objects
    *** VERSION - 3.1 LAST UPDATED - 1.1.2000
    *** DEVELOPER - Jason Holm
    *** COPYRIGHT - Ingenious Student Labs
    *** DATE CREATED - 1.1.2000
    ***
RemEnd
REM *** START SYSTEM SETUP SECTION

sync on : sync rate 30
autocam off
hide mouse
randomize timer()

REM *** STOP SYSTEM SETUP SECTION
REM ****************************************

REM ****************************************
REM *** START INTRO SECTION
REM *** INTRO SECTION HEADER

REM DECLARE VARIABLES
1pX# = 50
1pZ# = 50
1aY = 0
REM SCREEN DISPLAY
cls
load bitmap "images/cubetitle.bmp"
ink rgb(255,255,255),0 : set text font "arial" : set text size 14 :
set text to normal : set text transparent
center text 320,460,"Copyright (c) 2000 Ingenious Student Labs"
REM LOAD SOUNDS
REM SPECIAL EFFECTS
REM REFRESH SCREEN
sync

REM *** INTRO SECTION LOOP

do
   REM SCREEN DISPLAY
   REM CONTROL INPUT
   x=scancode() : if Keystate(x)=1 then goto MainSection
   REM REFRESH SCREEN
   sync
loop

REM *** END INTRO SECTION
```

```
REM ****************************************
REM ****************************************
REM *** START MAIN SECTION
REM *** MAIN SECTION HEADER

MainSection:
REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls
backdrop off
REM OBJECT CREATION
make object cube 1,200
   position object 1,1pX#,0,1pZ#
   rotate object 1, 0,1aY,0
make object cylinder 3, 100 : Rem Make Object 3 a cylinder marking
the X axis
   color object 3,rgb(255,0,0)
   scale object 3, 5,1000,5
   position object 3, 1pX#,0,1pZ#
   set object rotation zyx 3
   rotate object 3, 1aY,0,90
make object cylinder 4, 100 : Rem Make Object 4 a cylinder marking
the Y axis
   color object 4,rgb(0,255,0)
   scale object 4, 5,1000,5
   position object 4, 1pX#,0,1pZ#
   rotate object 4, 0,1ay,0
make object cylinder 5, 100 : Rem Make Object 5 a cylinder marking
the Z axis
   color object 5,rgb(0,0,255)
   scale object 5, 5,1000,5
   position object 5, 1pX#,0,1pZ#
   set object rotation zyx 3
   rotate object 5, 90,180,1aY
REM LOAD MODELS
REM SET CAMERA
position camera -20,200,-500
point camera 0,50,0
REM REFRESH SCREEN
sync

REM *** INPUT SECTION

ink rgb(255,255,255),0 : print "The cube is currently 200 units high
by 200 units wide by 200 units deep."
do
   if scancode()=0 then exit
   sync
loop
```

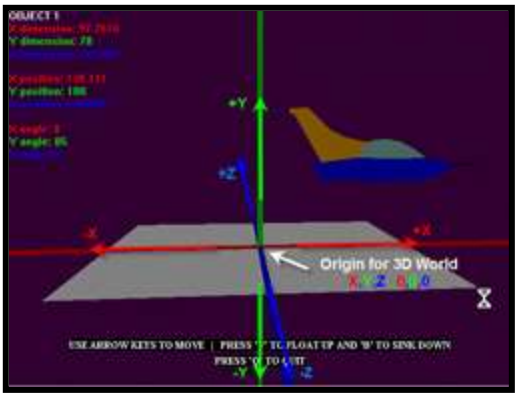
```
ink rgb(255,0,0),0 : input "Increase/Decrease X by (percent):
";1percentX#
ink rgb(0,255,0),0 : input "Increase/Decrease Y by (percent):
";1percentY#
ink rgb(0,0,255),0 : input "Increase/Decrease Z by (percent):
";1percentZ#
do
   if scancode()=0 then exit
   sync
loop
1dX# = (1percentX# / 100) * object size x(1)
1dY# = (1percentY# / 100) * object size y(1)
1dZ# = (1percentZ# / 100) * object size z(1)
scale object 1, 1percentX#,1percentY#,1percentZ#
backdrop on

REM *** MAIN SECTION LOOP

do
   REM SPECIAL EFFECTS
   REM OBJECT ORIENTATIONS
   1pX# = object position X(1)
   1pZ# = object position Z(1)
   1aY = object angle Y(1)
   REM CAMERA ORIENTATIONS
   REM LIVE SCREEN DISPLAY
   ink rgb(255,255,255),0 : set text font "times" : set text size 16
: set text to bold : set text transparent
   center text 320,420,"USE ARROW KEYS TO MOVE   |   PRESS SPACE BAR
TO CHANGE COLOR"
   center text 320,440,"PRESS 'R' TO RESIZE   |   PRESS 'Q' TO QUIT"
   set cursor 0,0
   print "OBJECT 1:"
   ink rgb(255,0,0),0 : print " X position: " ; 1pX#
   ink rgb(0,0,255),0 : print " Z position: " ; 1pZ#
   print
   ink rgb(0,255,0),0 : print " Y angle: " ; 1aY
   print
   ink rgb(255,255,255),0 : print "DIMENSIONS:"
   ink rgb(255,0,0),0 : print " Original X: ";object size x(1)
   print " Scale X Percentage: ";1percentX#
   print " Scaled X: ";1dX#
   print
   ink rgb(0,255,0),0 : print " Original Y: ";object size y(1)
   print " Scale Y Percentage: ";1percentY#
   print " Scaled Y: ";1dY#
   print
   ink rgb(0,0,255),0 : print " Original Z: ";object size z(1)
   print " Scale Z Percentage: ";1percentZ#
   print " Scaled Z: ";1dZ#
```

```
   REM CONTROL INPUT
   if Upkey()=1 then move object 1,10
   if Downkey()=1 then move object 1,-10
   if Leftkey()=1 then 1aY = wrapvalue(1aY-5)
   if Rightkey()=1 then 1aY = wrapvalue(1aY+5)
   if Inkey$()="q"
      delete object 1 : Rem Delete Object 1 from the game field
      for x = 3 to 5
         delete object x
      next x
      backdrop off
      1pX# = 50
      1pZ# = 50
      1aY = 0
      goto EndSection
   endif
   if Inkey$()="r"
      delete object 1 : Rem Delete Object 1 from the game field
      for x = 3 to 5
         delete object x
      next x
      backdrop off
      goto MainSection
   endif
   red=rnd(255) : green=rnd(255) : blue=rnd(255)
   if Spacekey()=1 then color object 1,rgb(red,green,blue)
   REM TRANSFORM OBJECTS
   REM MOVE OBJECTS
   Yrotate object 1,1aY

   Xrotate object 3,1aY
   Yrotate object 4,1aY
   Zrotate object 5,1aY

   for x = 3 to 5
      position object x,1pX#,0,1pZ#
   next x
   REM CHECK FOR COLLISION
   REM REFRESH SCREEN
   sync
loop

REM *** STOP MAIN SECTION
REM ***************************************
```

```
REM ****************************************
REM *** START END SECTION
REM *** END SECTION HEADER

EndSection:
REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls 0
ink rgb(255,255,255),0 : set text font "times" : set text size 20 :
set text to bold : set text transparent
center text 320,240,"PLAY AGAIN [Y/N]?"
REM LOAD SOUNDS
REM SPECIAL EFFECTS
REM REFRESH SCREEN
sync

REM *** END SECTION LOOP

do
   REM CONTROL INPUT
   if Inkey$()="y" then goto MainSection
   if Inkey$()="n"
      cls : end
   endif
   REM REFRESH SCREEN
   sync
loop

end
REM *** STOP END SECTION
REM ****************************************
```

The 3D World

You have seen how 3D objects take up 3D space and how you can change the size of the objects by manipulating the object's scale along one of its 3 axes (x, y, z). Now let's look at how object's position and movement is defined in the 3D world. All positions and motions are described as relative to the origin of the 3D world. The point whose values are 0,0,0 is called the origin.



The X,Y,Z value of the point and whether the values are negative or positive shows you the position of the point relative to the origin.

WORKSHEET

Load in certification3_2.dba and run it. Sketch the object and each axis and carefully title the X,Y,Z axes in the certification3_2.dba game.

certification3_2.dba

```
REM ****************************************
RemStart
    ***
    *** TITLE - 3D Space
    *** VERSION - 3.2 LAST UPDATED - 1.1.2000
    *** DEVELOPER - Jason Holm
    *** COPYRIGHT - Ingenious Student Labs
    *** DATE CREATED - 1.1.2000
    ***
RemEnd
REM *** START SYSTEM SETUP SECTION

sync on : sync rate 30
autocam off
hide mouse
randomize timer()

REM *** STOP SYSTEM SETUP SECTION
REM ****************************************

REM ****************************************
REM *** START INTRO SECTION
REM *** INTRO SECTION HEADER

REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls
load bitmap "images/cubetitle.bmp"
ink rgb(255,255,255),0 : set text font "arial" : set text size 14 :
set text to normal : set text transparent
center text 320,460,"Copyright (c) 2000 Ingenious Student Labs"
REM LOAD SOUNDS
REM SPECIAL EFFECTS
REM REFRESH SCREEN
sync

REM *** INTRO SECTION LOOP

do
   REM SCREEN DISPLAY
   REM CONTROL INPUT
   x=scancode() : if Keystate(x)=1 then goto MainSection
   REM REFRESH SCREEN
   sync
loop

REM *** END INTRO SECTION
REM ****************************************
```

```
REM ****************************************
REM *** START MAIN SECTION
REM *** MAIN SECTION HEADER

MainSection:
REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls
backdrop on : Rem Create the 3D world's background
color backdrop rgb(50,0,50) : Rem Change the Backdrop color
REM OBJECT CREATION
make object cube 1, 50
   color object 1,rgb(255,0,0)
   1dX# = 50 : Rem The X size you want the Object to become
   1dY# = 120 : Rem The Y size you want the Object to become
   1dZ# = 30 : Rem The Z size you want the Object to become
   scale object 1, (1dX# / object size x(1))*100,(1dY# / object size
y(1))*100,(1dZ# / object size z(1))*100
   remStart
   Since the default size of this cube is 50 in each direction, this
function becomes
      scale object 1, (50 / 50) * 100, (120 / 50) * 100, (30 / 50) *
100
      scale object 1, (1) * 100, (2.4) * 100, (.6) * 100
      scale object 1, 100,240,60
   remEnd
   position object 1, 0,0,0 : Rem Place Object 1 at a specific place
in the world
   rotate object 1,0,350,0 : Rem Rotate Object 1 a specific amount
make object plain 2, 300,200 : Rem Make Object 2 a flat plane
   rotate object 2, 90,0,0
make object cylinder 3, 100 : Rem Make Object 3 a cylinder marking
the X axis
   color object 3,rgb(255,0,0)
   scale object 3, 5,1000,5
   position object 3, 0,0,0
   rotate object 3, 0,0,90
make object cylinder 4, 100 : Rem Make Object 4 a cylinder marking
the Y axis
   color object 4,rgb(0,255,0)
   scale object 4, 5,1000,5
   position object 4, 0,0,0
   rotate object 4, 0,90,0
make object cylinder 5, 100 : Rem Make Object 5 a cylinder marking
the Z axis
   color object 5,rgb(0,0,255)
   scale object 5, 5,5000,5
   position object 5, 0,0,0
   rotate object 5, 90,0,0
REM LOAD MODELS
```

```
REM SET CAMERA
position camera -20,100,-300
point camera 0,50,0
REM REFRESH SCREEN
sync

REM *** MAIN SECTION LOOP

do
   REM SPECIAL EFFECTS
   REM OBJECT ORIENTATIONS
   1pX# = object position X(1)
   1pY# = object position Y(1)
   1pZ# = object position Z(1)
   1aX = object angle X(1)
   1aY = object angle Y(1)
   1aZ = object angle Z(1)
   REM CAMERA ORIENTATIONS
   REM LIVE SCREEN DISPLAY
   ink rgb(255,255,255),0 : set text font "times" : set text size 16
: set text to bold : set text opaque
   center text 320,420,"USE ARROW KEYS TO MOVE    |    PRESS 'V' TO
FLOAT UP AND 'B' TO SINK DOWN"
   center text 320,440,"PRESS 'Q' TO QUIT"
   Rem Print Object Dimensions
   set cursor 0,0
   ink rgb(255,255,255),0 : print "OBJECT 1"
   ink rgb(255,0,0),0 : print "X dimension: ";1dX#
   ink rgb(0,255,0),0 : print "Y dimension: ";1dY#
   ink rgb(0,0,255),0 : print "Z dimension: ";1dZ#
   print
   Rem Print Object position
   ink rgb(255,0,0),0 : print "X position: ";1pX#
   ink rgb(0,255,0),0 : print "Y position: ";1pY#
   ink rgb(0,0,255),0 : print "Z position: ";1pZ#
   print
   Rem Print Object angles
   ink rgb(255,0,0),0 : print "X angle: ";1aX
   ink rgb(0,255,0),0 : print "Y angle: ";1aY
   ink rgb(0,0,255),0 : print "Z angle: ";1aZ
   REM CONTROL INPUT
   if Upkey()=1 then move object 1,5
   if Downkey()=1 then move object 1,-5
   if Leftkey()=1 then Yrotate object 1,wrapvalue(1aY-5)
   if Rightkey()=1 then Yrotate object 1,wrapvalue(1aY+5)
   if Inkey$()="v" then 1pY# = 1pY# + 5 : Rem If the user hits the
'v' key, then move Object 1 up 5 units
   if Inkey$()="b" then 1pY# = 1pY# - 5 : Rem If the user hits the
'b' key, then move Object 1 down 5 units
```

```
   if Inkey$()="q"
      for x = 1 to 100
         if object exist(x) = 1
            delete object x
         endif
      next x
      backdrop off : Rem Remove the 3D world's background
      goto EndSection
   endif
   REM TRANSFORM OBJECTS
   REM MOVE OBJECTS
   Rem Store the new positions of Object 1
   1pX# = object position X(1)
   1pZ# = object position Z(1)
   position object 1,1pX#,1pY#,1pZ# : Rem Reposition Object 1
   REM CHECK FOR COLLISION
   REM REFRESH SCREEN
   sync
loop

REM *** STOP MAIN SECTION
REM ***************************************

REM ***************************************
REM *** START END SECTION
REM *** END SECTION HEADER

EndSection:
REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls 0
ink rgb(255,255,255),0 : set text font "times" : set text size 20 :
set text to bold : set text transparent
center text 320,240,"PLAY AGAIN [Y/N]?"
REM LOAD SOUNDS
REM SPECIAL EFFECTS
REM REFRESH SCREEN
sync

REM *** END SECTION LOOP

do
   REM CONTROL INPUT
   if Inkey$()="y" then goto MainSection
   if Inkey$()="n"
      cls : end
   endif
   REM REFRESH SCREEN
   sync
loop

end
REM *** STOP END SECTION
REM ***************************************
```

Your Action:
Load and run ***certification3_2-demo.dba***. Move the space ship around and note the change in the X,Y, Z values. Does the X value change as the space ship moves away from the screen? What value changes to a negative as the ship descends below the origin point?

WORKSHEET

Use the table below to experiment with position and direction in the 3D world. Using the certification3_2demo.dba file, move the ship to the positions shown below and then describe the position and direction relative to the origin.

x=100 | y=150 | z=300 | y angle=180
The ship is right, back and above the origin. The ship is facing toward the player.

x=100 | y=-45 | z=-100 | y angle: 225
The ship is left, forward, and below the origin. The ship is facing towards the player and slightly to the left.

x=100 | y=100 | z=-100 | y angle: 90
T*he ship is right, forward, and above the origin. The ship is facing right.*

x=-150 | y=-225 | z=240 | y angle: 270
The ship is left, back, and below the origin. The ship is facing left.

certification3_2-demo.dba

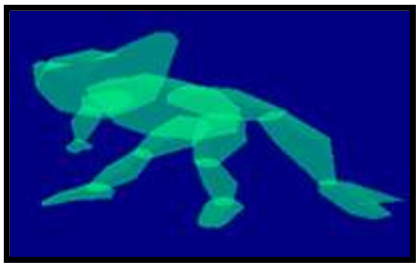
```
REM ****************************************
RemStart
    ***
    *** TITLE - 3D Spaceship
    *** VERSION - 3.2 LAST UPDATED - 1.1.2000
    *** DEVELOPER - Jason Holm
    *** COPYRIGHT - Ingenious Student Labs
    *** DATE CREATED - 1.1.2000
    ***
RemEnd
REM *** START SYSTEM SETUP SECTION

sync on : sync rate 30
autocam off
hide mouse
randomize timer()

REM *** STOP SYSTEM SETUP SECTION
REM ****************************************

REM ****************************************
REM *** START INTRO SECTION
REM *** INTRO SECTION HEADER

REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls
load bitmap "images/cubetitle.bmp"
ink rgb(255,255,255),0 : set text font "arial" : set text size 14 :
set text to normal : set text transparent
center text 320,460,"Copyright (c) 2000 Ingenious Student Labs"
REM LOAD SOUNDS
REM SPECIAL EFFECTS
REM REFRESH SCREEN
sync

REM *** INTRO SECTION LOOP

do
   REM SCREEN DISPLAY
   REM CONTROL INPUT
   x=scancode() : if Keystate(x)=1 then goto MainSection
   REM REFRESH SCREEN
   sync
loop

REM *** END INTRO SECTION
REM ****************************************
```

```
REM ****************************************
REM *** START MAIN SECTION
REM *** MAIN SECTION HEADER

MainSection:
REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls
backdrop on : Rem Create the 3D world's background
color backdrop rgb(50,0,50) : Rem Change the Backdrop color
REM OBJECT CREATION
make object plain 2, 300,200 : Rem Make Object 2 a flat plane
   rotate object 2, 90,0,0
make object cylinder 3, 100 : Rem Make Object 3 a cylinder marking
the X axis
   color object 3,rgb(255,0,0)
   position object 3, 0,0,0
   scale object 3, 5,1000,5
   rotate object 3, 0,0,90
make object cylinder 4, 100 : Rem Make Object 4 a cylinder marking
the Y axis
   color object 4,rgb(0,255,0)
   position object 4, 0,0,0
   scale object 4, 5,1000,5
   rotate object 4, 0,90,0
make object cylinder 5, 100 : Rem Make Object 5 a cylinder marking
the Z axis
   color object 5,rgb(0,0,255)
   position object 5, 0,0,0
   scale object 5, 5,5000,5
   rotate object 5, 90,0,0
REM LOAD MODELS
load object "models/spcveh20.x",1
   Yrotate Object 1,180
   fix object pivot 1
      Rem Resize Object 8
      1dY = 70

      1scale# = (1dY / object size y(1))*100
      scale object 1, 1scale#, 1scale#, 1scale#
      1dX# = (1scale# / 100) * object size x(1)
      1dY# = (1scale# / 100) * object size y(1)
      1dZ# = (1scale# / 100) * object size z(1)

   position object 1,0,1dY#/2,0
   rotate object 1,0,0,0
REM SET CAMERA
position camera -20,100,-300
point camera 0,50,0
REM REFRESH SCREEN
sync
```

```
REM *** MAIN SECTION LOOP

do
   REM SPECIAL EFFECTS
   REM OBJECT ORIENTATIONS
   1pX# = object position X(1)
   1pY# = object position Y(1)
   1pZ# = object position Z(1)
   1aX = object angle X(1)
   1aY = object angle Y(1)
   1aZ = object angle Z(1)
   REM CAMERA ORIENTATIONS
   REM LIVE SCREEN DISPLAY
   ink rgb(255,255,255),0 : set text font "times" : set text size 16
: set text to bold : set text opaque
   center text 320,420,"USE ARROW KEYS TO MOVE    |    PRESS 'V' TO
FLOAT UP AND 'B' TO SINK DOWN"
   center text 320,440,"PRESS 'Q' TO QUIT"
   Rem Print Object Dimensions
   set cursor 0,0
   ink rgb(255,255,255),0 : print "OBJECT 1"
   ink rgb(255,0,0),0 : print "X dimension: ";1dX#
   ink rgb(0,255,0),0 : print "Y dimension: ";1dY#
   ink rgb(0,0,255),0 : print "Z dimension: ";1dZ#
   print
   Rem Print Object position
   ink rgb(255,0,0),0 : print "X position: ";1pX#
   ink rgb(0,255,0),0 : print "Y position: ";1pY#
   ink rgb(0,0,255),0 : print "Z position: ";1pZ#
   Rem Print Object angles
   print
   ink rgb(255,0,0),0 : print "X angle: ";1aX
   ink rgb(0,255,0),0 : print "Y angle: ";1aY
   ink rgb(0,0,255),0 : print "Z angle: ";1aZ
   REM CONTROL INPUT
   if Upkey()=1 then move object 1,10
   if Downkey()=1 then move object 1,-10
   if Leftkey()=1 then 1aY = wrapvalue(1aY-5)
   if Rightkey()=1 then 1aY = wrapvalue(1aY+5)
   if Inkey$()="v" then 1pY# = 1pY# + 5 : Rem If the user hits the
'v' key, then move Object 1 up 5 units
   if Inkey$()="b" then 1pY# = 1pY# - 5 : Rem If the user hits the
'b' key, then move Object 1 down 5 units
   if Inkey$()="q"
      for x = 1 to 5
         delete object x : Rem Then delete Objects 1 to 5 from the
game field
      next x
      backdrop off : Rem Then remove the 3D world's background
      goto EndSection
   endif
```

```
   REM TRANSFORM OBJECTS
   REM MOVE OBJECTS
   Rem Store the new positions of Object 1
   1pX# = object position X(1)
   1pZ# = object position Z(1)
   Yrotate object 1,1aY
   position object 1,1pX#,1pY#,1pZ# : Rem Reposition Object 1
   REM CHECK FOR COLLISION
   REM REFRESH SCREEN
   sync
loop

REM *** STOP MAIN SECTION
REM ************************************

REM ************************************
REM *** START END SECTION
REM *** END SECTION HEADER

EndSection:
REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls 0
ink rgb(255,255,255),0 : set text font "times" : set text size 20 :
set text to bold : set text transparent
center text 320,240,"PLAY AGAIN [Y/N]?"
REM LOAD SOUNDS
REM SPECIAL EFFECTS
REM REFRESH SCREEN
sync

REM *** END SECTION LOOP

do
   REM CONTROL INPUT
   if Inkey$()="y" then goto MainSection
   if Inkey$()="n"
      cls : end
   endif
   REM REFRESH SCREEN
   sync
loop

end
REM *** STOP END SECTION
REM ************************************
```

Animated Objects



To make a game more interesting, you will want some of your objects to have motion as they move...possibly to simulate walking, flying (propeller turning), or swinging a samurai sword. The way game programmers create the impression of action is with animated characters. Typically in the game world, if you wanted to create a skate boarder character, then you would create an animated model that would have the skater in a number of different positions and attitudes. As the player hit the correct inputs to create a jump, ollie or a grind, then the program would simply switch and display that short animation model.

Let's look at a simple animated model, of a goblin that is walking. Because we are working in a 3D game environment, you can view the animated model from any direction (the model was created in 3D). But the action (animation) is only of walking.

VIDEO: Animated Objects (3:16)

Your Action:
Load and run ***certification3_walk.dba***. Observe how the object moves.

WORKSHEET
What is the command for loading in a pre-made model to be used in a game?
load object "walk.x",1

What is the big challenge of creating a walking object?
Making sure the action is smooth. A walk animation is looped. If the loop/animation doesn't begin and end exactly the same way, you will notice the character jerking every step.

certification3_walk.dba

```
Rem * Title  : Managing Models
Rem * Author : DBS-LB
Rem * Date   : 1st Sept 99
rem ========================================
rem DARK BASIC EXAMPLE PROGRAM 2
rem ========================================
rem This program manages your models features
rem ----------------------------------------

rem Load a bitmap and grab image
load bitmap "images/floor1.bmp",1
get image 1,0,0,128,128
delete bitmap 1

rem Load your object
load object "models/walk.x",1

rem Rotate and fix data so character faces right way
xrotate object 1,0
yrotate object 1,180
zrotate object 1,0
fix object pivot 1

rem Immediately hide the object
hide object 1

rem Position your object 100 units into the distance
position object 1,0,0,100

rem Rotate your object to face an angle of 45 degrees
rotate object 1,0,45,0

rem Scale your object to double its width and depth, but not height
scale object 1,200,100,200

rem Texture your entire object using the cloth image grabbed and
stored in image 1
texture object 1,1

rem Color the object
color object 1,rgb(0,255,0)

rem Ghost your object to make it semi-transparent
ghost object on 1

rem Fade object to 25 percent intensity
fade object 1,25

rem Animate your object (loop to frame 25 and back to 5)
loop object 1,5,25
```

```
rem Show the object
show object 1

rem Lock and unlock the object
lock object on 1
lock object off 1

rem Reverse rotation
set object rotation xyz 1

rem Begin main loop
sync on
draw to front
while mouseclick()=0

rem Show FPS
set cursor 0,0
print screen fps()

rem Control the object movement with the cursor keys
if upkey()=1 then move object 1,1.0
if downkey()=1 then move object 1,-1.0

rem Control the object rotation with the cursor keys
angle#=object angle y(1)
if leftkey()=1 then angle#=wrapvalue(angle#-2)
if rightkey()=1 then angle#=wrapvalue(angle#+2)
yrotate object 1,angle#

rem Use the spacebar to point your object to the center of the world
if spacekey()=1 then point object 1,0,0,0

rem Synchronize
sync

rem Set rotation order back to normal
set object rotation zyx 1

rem End main loop
endwhile
draw to back

rem Stop your object from animating
stop object 1

rem Switch off ghosting
ghost object off 1

rem Free your object from memory
delete object 1

rem End the program
end
```

The challenge of creating animated models

Did you notice how the walking action looped or repeated itself every few steps? This is the challenge of creating animations for action games. For things like walking or running, you need to create an animation where the end and the beginning are exactly the same. The next time you play a game, watch to see how the characters walk or run. What you are seeing is the same small one-step animation repeated hundreds of times (maybe thousands) to make it look like one long walk or run.

Multiple objects

The one thing you must not forget about using multiple objects is the need for a unique ID for each object. In games you will have objects that are part of the background. They may be random or grouped together (forest of trees, rocks in a desert, clouds in the sky); you might choose to use a loop to randomly generate and distribute the objects. Basically, a game with one or two objects is much easier to create than a game with hundreds or thousands.

VIDEO: Multiple Objects (4:19)

Your Action:
Load in ***certification3_3.dba***. Look at the multiple objects in the code, then run the program and identify the objects. Take note of the color of each of the cylinders and its ID number so you can tell which object is which when you run the game.

WORKSHEET
List the 5 objects in certification3_3.dba, along with their object ID number and color.

Object	Object ID Number	Object Color
Cube	*1*	*255,0,0 (Red)*
Cylinder	*3*	*255,0,0 (Red)*
Cylinder	*4*	*0,255,0 (Green)*
Cylinder	*5*	*0,0,255 (Blue)*
Cube	*6*	*0,0,255 (Blue)*

certification3_3.dba

```
REM ****************************************
RemStart
    ***
    *** TITLE - Multiple Objects
    *** VERSION - 3.3 LAST UPDATED - 1.1.2000
    *** DEVELOPER - Jason Holm
    *** COPYRIGHT - Ingenious Student Labs
    *** DATE CREATED - 1.1.2000
    ***
RemEnd
REM *** START SYSTEM SETUP SECTION

sync on : sync rate 30
autocam off
hide mouse
randomize timer()

REM *** STOP SYSTEM SETUP SECTION
REM ****************************************

REM ****************************************
REM *** START INTRO SECTION
REM *** INTRO SECTION HEADER

REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls
load bitmap "images/cubetitle.bmp"
ink rgb(255,255,255),0 : set text font "arial" : set text size 14 :
set text to normal : set text transparent
center text 320,460,"Copyright (c) 2000 Ingenious Student Labs"
REM LOAD SOUNDS
REM SPECIAL EFFECTS
REM REFRESH SCREEN
sync

REM *** INTRO SECTION LOOP

do
   REM SCREEN DISPLAY
   REM CONTROL INPUT
   x=scancode() : if Keystate(x)=1 then goto MainSection
   REM REFRESH SCREEN
   sync
loop

REM *** END INTRO SECTION
REM ****************************************
```

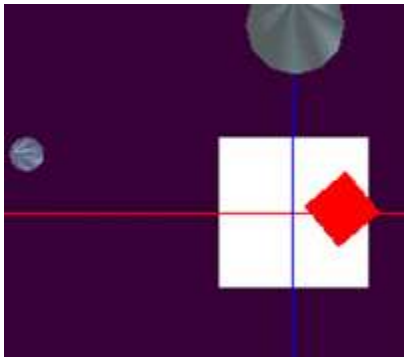
```
REM ****************************************
REM *** START MAIN SECTION
REM *** MAIN SECTION HEADER

MainSection:
REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls
backdrop on
color backdrop rgb(50,0,50)
REM OBJECT CREATION ⬅ Objects are created in this section
make object cube 1, 70 : Rem Object 1 : The Red Cube
   color object 1,rgb(255,0,0)
   position object 1, 0,35,0
make object plain 2, 300,200 : Rem Object 2 : The Ground
   rotate object 2, 90,0,0
make object cylinder 3, 100 : Rem Object 3 : The X Axis
   color object 3,rgb(255,0,0)
   scale object 3, 5,1000,5
   position object 3, 0,0,0
   rotate object 3, 0,0,90
make object cylinder 4, 100 : Rem Object 4 : The Y Axis
   color object 4,rgb(0,255,0)
   scale object 4, 5,1000,5
   position object 4, 0,0,0
   rotate object 4, 0,90,0
make object cylinder 5, 100 : Rem Object 5 : The Z Axis
   color object 5,rgb(0,0,255)
   scale object 5, 5,5000,5
   position object 5, 0,0,0
   rotate object 5, 90,0,0
make object cube 6,70 : Rem Object 5 : The Blue Cube
   color object 6,rgb(0,0,255)
   position object 6, -150,35,150
NumberOfCones=20 : Rem Choose your number of cones
for x=10 to NumberOfCones+10 : Rem Create a series of random cones,
numbered Objects 10 to the number of cones plus 10
   make object cone x, rnd(100)+50 : Rem Make each cone a random
height from 50 to 150
   color object x,rgb(rnd(105)+150,rnd(105)+150,rnd(105)+150) : Rem
Make each cone a random color, but make sure the color is not too
dark
```

```
   do
      NoCenterX = rnd(1500)-750 : Rem Make a random X coordinate,
from -750 to 750
      NoCenterZ = rnd(1500)-750 : Rem Make a random Z coordinate,
from -750 to 750
      if (NoCenterX > -150 and NoCenterX < 150  and NoCenterZ > -150
and NoCenterZ < 150) = 0 and NoCenterZ > 150
      Rem This checks to make sure the cones aren't created too close
to the cube's starting point (or behind the viewing area)
         exit
         Rem If they aren't, the program continues
      endif
      Rem If they are, they are sent back to choose a new random
location
   loop
   position object x, NoCenterX,object size(x)*.75,NoCenterZ
   remStart
      Place each cone at a random spot, but make sure they are all:
      1: Anywhere in a 1500 by 1500 unit square around the start
point
      2: Not in the 300 by 300 square right in the middle
      3: In front of the camera
      4: Flat on the ground by taking its height, (which is one and
one-half the diameter)
         multiplying it by 3/4 (.75), and raising it that amount
   remEnd
next x : Rem Create the next cone
   magic_cone = rnd(NumberOfCones)+11 : Rem Randomly declare one of
the cones a 'magic' cone
REM LOAD MODELS
REM SET CAMERA
position camera -20,100,-300
point camera 0,50,0
REM REFRESH SCREEN
sync

REM *** MAIN SECTION LOOP

do
   REM SPECIAL EFFECTS
   for pulse = 0 to 5
   REM OBJECT ORIENTATIONS
   Rem Object 1 Variables
   1pX# = object position X(1)
   1pZ# = object position Z(1)
   1aY = object angle Y(1)
   Rem Object 6 Variables
   6pX# = object position X(6)
   6pZ# = object position Z(6)
   6aY = object angle Y(6)
   REM CAMERA ORIENTATIONS
```

```
   REM LIVE SCREEN DISPLAY
   ink rgb(255,255,255),0 : set text font "times" : set text size 16
: set text to bold : set text opaque
   center text 320,420,"USE ARROW KEYS TO MOVE THE RED CUBE   |   USE
THE NUMBER PAD TO MOVE THE BLUE CUBE"
   center text 320,440,"PRESS 'Q' TO QUIT"
   REM CONTROL INPUT
   Rem Object 1 Controls
   if Upkey()=1 then move object 1,10
   if Downkey()=1 then move object 1,-10
   if Leftkey()=1 then Yrotate object 1,wrapvalue(1aY-5)
   if Rightkey()=1 then Yrotate object 1,wrapvalue(1aY+5)
   Rem Object 1 Controls
   if Inkey$()="8" then move object 6,10
   if Inkey$()="2" then move object 6,-10
   if Inkey$()="4" then Yrotate object 6,wrapvalue(6aY-5)
   if Inkey$()="6" then Yrotate object 6,wrapvalue(6aY+5)

   if Inkey$()="q"
      for x = 1 to 100
         if object exist(x) = 1 : Rem Check to see if each object
exists
            delete object x : Rem if it does, delete it from the game
field
         endif
      next x
      backdrop off
      goto EndSection
   endif
   REM TRANSFORM OBJECTS
   size = (pulse*20)+100 : Rem Make Objects pulse by growing and
contracting
   scale object magic_cone,size,size,size : Rem Make the 'magic' cone
pulse
   REM MOVE OBJECTS
   REM CHECK FOR COLLISION
   REM REFRESH SCREEN
   sync
   next pulse
loop

REM *** STOP MAIN SECTION
REM ***************************************
```

```
REM ****************************************
REM *** START END SECTION
REM *** END SECTION HEADER

EndSection:
REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls 0
ink rgb(255,255,255),0 : set text font "times" : set text size 20 :
set text to bold : set text transparent
center text 320,240,"PLAY AGAIN [Y/N]?"
REM LOAD SOUNDS
REM SPECIAL EFFECTS
REM REFRESH SCREEN
sync

REM *** END SECTION LOOP

do
   REM CONTROL INPUT
   if Inkey$()="y" then goto MainSection
   if Inkey$()="n"
      cls : end
   endif
   REM REFRESH SCREEN
   sync
loop

end
REM *** STOP END SECTION
REM ****************************************
```

View / Camera



Cameras allow you to give the player a custom or unique view of the game's 3D world. You can have a camera focus from one spot, or the camera can follow the action from within the cockpit of a space fighter or just behind an extreme athlete. Camera views can allow the game player to have a more realistic experience, allowing the gamer to experience the game from the center of the action, or possibly a view from on high looking down upon the peasants as they work the fields owned by the Dark Knight.

Now let's look at a couple ways cameras are used in a game:

VIDEO: Cameras in a Game (4:32)

Your Action:
Load in ***certification3_4.dba*** and change where cameras 1 and 2 point.

WORKSHEET
FOLLOW ME = 1 is the variable used by the if/then statement to have the camera do what?
Tells the camera to follow object 1

If you wanted to position a camera high above and look down at the center of the 3D world, how would you complete the following code?
POSITION CAMERA *0,1500,0*
POINT CAMERA *0,0,0*

certification3_4.dba

```
REM ****************************************
RemStart
    ***
    *** TITLE - Cameras
    *** VERSION - 3.4 LAST UPDATED - 1.1.2000
    *** DEVELOPER - Jason Holm
    *** COPYRIGHT - Ingenious Student Labs
    *** DATE CREATED - 1.1.2000
    ***
RemEnd
REM *** START SYSTEM SETUP SECTION

sync on : sync rate 30
autocam off : Rem Set autocamera options
hide mouse
randomize timer()

REM *** STOP SYSTEM SETUP SECTION
REM ****************************************

REM ****************************************
REM *** START INTRO SECTION
REM *** INTRO SECTION HEADER

REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls 0
load bitmap "images/cubetitle.bmp"
ink rgb(255,255,255),0 : set text font "arial" : set text size 14 :
set text to normal : set text transparent
center text 320,460,"Copyright (c) 2000 Ingenious Student Labs"
REM LOAD SOUNDS
REM SPECIAL EFFECTS
REM REFRESH SCREEN
sync

REM *** INTRO SECTION LOOP

do
   REM SCREEN DISPLAY
   REM CONTROL INPUT
   x=scancode() : if Keystate(x)=1 then goto MainSection
   REM REFRESH SCREEN
   sync
loop

REM *** END INTRO SECTION
REM ****************************************
```

```
REM ****************************************
REM *** START MAIN SECTION
REM *** MAIN SECTION HEADER

MainSection:
REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls 0
backdrop on
color backdrop rgb(50,0,50)
REM OBJECT CREATION
make object cube 1, 100 : Rem Cube
   color object 1,rgb(255,0,0)
   position object 1, 0,50,0
make object plain 2, 300,300 : Rem Ground
   rotate object 2, 90,0,0
make object cylinder 3, 100 : Rem X axis
   color object 3,rgb(255,0,0)
   position object 3, 0,0,0
   scale object 3, 5,5000,5
   rotate object 3, 0,0,90
make object cylinder 4, 100 : Rem Y axis
   color object 4,rgb(0,255,0)
   position object 4, 0,0,0
   scale object 4, 5,5000,5
   rotate object 4, 0,90,0
make object cylinder 5, 100 : Rem Z axis
   color object 5,rgb(0,0,255)
   position object 5, 0,0,0
   scale object 5, 5,5000,5
   rotate object 5, 90,0,0
NumberOfCones=20 : NC = NumberOfCones + 10
for x=10 to NC : Rem Cones
   make object cone x, rnd(200)+50
   color object x,rgb(rnd(105)+150,rnd(105)+150,rnd(105)+150)
   do
      NoCenterX = rnd(2000)-1000
      NoCenterZ = rnd(2000)-1000
      if (NoCenterX > -150 and NoCenterX < 150 and NoCenterZ > -150
and NoCenterZ < 150) = 0 then exit
   loop
   position object x, NoCenterX,object size(x)*.75,NoCenterZ
next x
REM LOAD MODELS
REM SET CAMERA ⬅ Change Camera settings here
position camera -20,100,-300 : Rem Set camera position
point camera 0,50,0 : Rem Rotate camera to point at a specific place
REM REFRESH SCREEN
sync
```

```
REM *** MAIN SECTION LOOP

do
   REM SPECIAL EFFECTS
   REM OBJECT ORIENTATIONS
   1pX# = object position X(1)
   1pY# = object position Y(1)
   1pZ# = object position Z(1)
   1aX = object angle X(1)
   1aY = object angle Y(1)
   1aZ = object angle Z(1)
   REM CAMERA ORIENTATIONS
   Rem Store the positions of the Camera as variables
   cpX# = camera position X()
   cpY# = camera position Y()
   cpZ# = camera position Z()
   Rem Store the angles of the Camera as variables
   caX# = camera angle X()
   caY# = camera angle Y()
   caZ# = camera angle Z()
   REM LIVE SCREEN DISPLAY
   ink 0,rgb(255,255,255) : set text font "times" : set text size 16
: set text to bold : set text opaque
   center text 320,420,"USE ARROW KEYS TO MOVE"
   center text 320,435,"PRESS [1] - [2] - [3] TO CHANGE CAMERA ANGLE"
   center text 320,450,"PRESS 'Q' TO QUIT"

   set cursor 0,0
   ink rgb(255,255,255),0 : print "OBJECT 1"
   Rem Print Object position
   ink rgb(255,0,0),0 : print "X position: ";1pX#
   ink rgb(0,255,0),0 : print "Y position: ";1pY#
   ink rgb(0,0,255),0 : print "Z position: ";1pZ#
   print
   Rem Print Object angles
   ink rgb(255,0,0),0 : print "X angle: ";1aX
   ink rgb(0,255,0),0 : print "Y angle: ";1aY
   ink rgb(0,0,255),0 : print "Z angle: ";1aZ

   Rem Print Camera positions
   set cursor 450,0 : ink rgb(255,255,255),0 : print "CAMERA"
   set cursor 450,16 : ink rgb(255,0,0),0 : print "Camera X position:
";cpX#
   set cursor 450,32 : ink rgb(0,255,0),0 : print "Camera Y position:
";cpY#
   set cursor 450,48 : ink rgb(0,0,255),0 : print "Camera Z position:
";cpZ#
   Rem Print Camera angles
   set cursor 450,80 : ink rgb(255,0,0),0 : print "Camera X angle:
";caX#
   set cursor 450,96 : ink rgb(0,255,0),0 : print "Camera Y angle:
";caY#
```

```
   set cursor 450,112 : ink rgb(0,0,255),0 : print "Camera Z angle:
";caZ#
   REM CONTROL INPUT
   if Upkey()=1 then move object 1,10
   if Downkey()=1 then move object 1,-10
   if Leftkey()=1 then Yrotate object 1,wrapvalue(1aY-5)
   if Rightkey()=1 then Yrotate object 1,wrapvalue(1aY+5)

   if Inkey$()="1"
      position camera -20,100,-300 : Rem Set camera position
      point camera 0,50,0 : Rem Rotate camera to point at a specific
place  ⬅ Camera settings
      followMe = 0 : Rem Tell the program not to follow Object 1
   endif

   if Inkey$()="2"
      position camera 0,1500,0 : Rem Set camera position
      point camera 0,0,0 : Rem Rotate camera to point at a specific
place  ⬅ Camera settings
      followMe = 0 : Rem Tell the program not to follow Object 1
   endif

   if Inkey$()="3"
      followMe = 1 : Rem Tell the program to follow Object 1
   endif

   if Inkey$()="q"
      for x = 1 to 100
         if object exist(x) = 1 then delete object x
      next x
      backdrop off
      goto EndSection
   endif
   REM TRANSFORM OBJECTS
   REM MOVE OBJECTS
   REM CHECK FOR COLLISION
   REM MOVE CAMERA
   if followMe = 1 : Rem If the camera is following Object 1
      cpZ# = Newzvalue(1pZ#,1aY-180,200) : Rem Calculate the camera's
new Z position
      cpX# = Newxvalue(1pX#,1aY-180,200) : Rem Calculate the camera's
new X position
      Position camera cpX#,200,cpZ# : Rem Set the camera placement
      Point camera 1pX#,1pY#+50,1pZ# : Rem Point camera at the top of
Object 1  ⬅ Camera settings
   endif
   REM REFRESH SCREEN
   sync
loop

REM *** STOP MAIN SECTION
REM ************************************
```

```
REM ****************************************
REM *** START END SECTION
REM *** END SECTION HEADER

EndSection:
REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls 0
ink rgb(255,255,255),0 : set text font "times" : set text size 20 :
set text to bold : set text transparent
center text 320,240,"PLAY AGAIN [Y/N]?"
REM LOAD SOUNDS
REM SPECIAL EFFECTS
REM REFRESH SCREEN
sync

REM *** END SECTION LOOP

do
   REM CONTROL INPUT
   if Inkey$()="y" then goto MainSection
   if Inkey$()="n"
      cls : end
   endif
   REM REFRESH SCREEN
   sync
loop

end
REM *** STOP END SECTION
REM ****************************************
```

Certification Module IV

Input, Collision, Texture, Sound, Packing and Final EXE

Certification IV Worksheet

Teacher's Version

When you played the Jetski game, what inputs made the game work?

UpArrow, LeftArrow, RightArrow (from jetski.dba)

What other controls might you have included for this game?

Answer may include start game, reset or restart game, end game

What is a collision?

A collision is when one object's space invades another object's space AND collision detection has been turned on for BOTH objects.
(found in certification text)

What commands would turn on and off the collision detection for an object with an ID of 2?

SET OBJECT COLLISION ON 2
SET OBJECT COLLISION OFF 2
(found in Collisions video)

In the Collision game certification4_2.dba, when you turned off the collision for object 7, what happened when the cube collided with the cone?

The cube passed through the cone.
(found in Collisions video)

What is the type of image file that can be used as a texture?

.bmp files only
(found in Textures video)

Why would you want to use textures?

Textures can make an object look more realistic.
(found in certification text)

Certification IV Worksheet

Teacher's Version

Where do you find the pre-made models and their textures in Dark Basic?

Under the menu selection MEDIA then under MEDIA BROWSER
(found in Texture Examples in Game Code video)

Describe a model that had more than one texture and the textures it used.

Any number of models under the MEDIA MEDIA BROWSER use multiple textures. One example is the Penguin that has 3 textures – body, leg and eye.

What type of sound files can you use with this game engine?

.wav and .mp3 files
(found in Sounds in Game Code video)

What is the command to load in a new sound?

Load sound "sounds/scifi5.wav",1
(found in Sounds in Game Code video)

What command will play a sound once?

Play sound 1
(found in Sounds in Game Code video)

What command will play a sound continuously?

Loop sound 7
(found in Sounds in Game Code video)

What command do you use to set the volume of sound 7 to a level of 50?

Set sound volume 7,50
(found in certification text)

What is the purpose of creating a final EXE?

It allows you to distribute the game.
(found in Making a Final EXE video)

Input - Human Touch

If you have ever played a video game and your player has died a horrible death, or if you lost your top score because you couldn't hit the correct key fast enough, then you have seen the big problem with interfacing the game to the human. We play games because they are fun, and the fun comes from playing a game. Watching a game play itself is probably only slightly less fun than watching snail races on a small black and white TV with no commercials. As a game designer, how you offer the player the chance to control the game is very important. The commands which are used the most need to be easy to perform. If you are designing a fast-action game, then you must make sure that the average user (and needless to say – buyer) of your video game will be able to control it.

Your Action:
Under ***myproj/jetski*** find and load in ***jetski.dba***. Run it and pay attention to the controls.

Notice how simple the controls are for the jetski game -- UP, RIGHT, and LEFT arrow keys.

WORKSHEET
When you played the Jetski game, what inputs made the game work?
UpArrow, LeftArrow, RightArrow

What other controls might you have included for this game?
Answer may include start game, reset or restart game, end game

VIDEO: Input from Humans (6:25)

Your Action:
Load in ***certfication4_1.dba***. Count the different inputs the game requires of the player and note what they are used for.

certification4_1.dba

```
REM ****************************************
RemStart
    ***
    *** TITLE - Inputs
    *** VERSION - 4.1 LAST UPDATED - 1.1.2000
    *** DEVELOPER - Jason Holm
    *** COPYRIGHT - Ingenious Student Labs
    *** DATE CREATED - 1.1.2000
    ***
RemEnd
REM *** START SYSTEM SETUP SECTION

sync on : sync rate 30
autocam off
hide mouse
randomize timer()

REM *** STOP SYSTEM SETUP SECTION
REM ****************************************

REM ****************************************
REM *** START INTRO SECTION
REM *** INTRO SECTION HEADER

REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls 0
load bitmap "images/cubetitle.bmp"
ink rgb(255,255,255),0 : set text font "arial" : set text size 14 :
set text to normal : set text transparent
center text 320,460,"Copyright (c) 2000 Ingenious Student Labs"
REM LOAD SOUNDS
REM SPECIAL EFFECTS
REM REFRESH SCREEN
sync

REM *** INTRO SECTION LOOP

do
   REM SCREEN DISPLAY
   REM CONTROL INPUT
      ⬇ Returns whatever key is being pressed right now (any key)
   x=scancode() : if Keystate(x)=1 then goto OptionsSection
                        ⬆ If the current key being pressed is x (any key), then this is true
   REM REFRESH SCREEN
   sync
loop

REM *** END INTRO SECTION
REM ****************************************
```

```
REM ****************************************
REM *** START OPTIONS SECTION
REM *** OPTIONS SECTION HEADER

OptionsSection:
REM DECLARE VARIABLES
   myInput$ = "Keyboard" : Rem Use the keyboard for input
   myName$ = "Anonymous" : Rem The default player name
REM SCREEN DISPLAY
cls 0
REM LOAD SOUNDS
REM SPECIAL EFFECTS
REM REFRESH SCREEN
sync

REM *** OPTIONS SECTION LOOP
do
   if scancode()=0 then exit
      ⬆ If no key is currently being pressed (waits for the user to release all keys)
loop
do
   REM SCREEN DISPLAY
   cls 0
   ink rgb(255,255,255),0 : set text font "times" : set text size 20
: set text to bold : set text transparent
   center text 320,140,"OPTIONS:"

   ink rgb(255,0,0),0 : center text 320,180,"PRESS 'N' TO ENTER YOUR
NAME"
   center text 320,200,"NAME: "+myName$ : Rem Display the name

   ink rgb(0,255,0),0 : center text 320,260,"PRESS 'K' TO USE ARROW
KEYS TO MOVE"
   center text 320,280,"PRESS 'M' TO USE MOUSE TO MOVE"
   center text 320,300,"PRESS 'B' TO USE BOTH KEYBOARD AND MOUSE TO
MOVE"
   center text 320,320,"INPUT: "+myInput$ : Rem Display the input
type

   ink rgb(0,0,255),0 : center text 320,360,"PRESS SPACE BAR TO PLAY"
```

```
   REM CONTROL INPUT
       ⬇ If the "n" key is currently pressed
   if Inkey$()="n" : Rem If the player presses 'n'
      do
         if Inkey$() <> "n" then exit : Rem Wait until they let go of
'n'          ⬆ Exit this loop when the "n" key is released
      loop
      ink rgb(255,0,0),0 : set cursor 200,220
      ⬇ Gather the characters typed by the user and save them to a variable
     input "ENTER NEW NAME: ";myName$ : Rem Wait for a response, then
record the response
   endif
       ⬇ If the "k" key is currently pressed
   if Inkey$()="k" then myInput$ = "Keyboard" : Rem If the player
hits 'k', change the variable myInput$ to "Keyboard"
       ⬇ If the "m" key is currently pressed
   if Inkey$()="m" then myInput$ = "Mouse" : Rem If the player hits
'm', change the variable myInput$ to "Mouse"
       ⬇ If the "b" key is currently pressed
   if Inkey$()="b" then myInput$ = "Both" : Rem If the player hits
'b', change the variable myInput$ to "Both"
       ⬇ If the Space Bar is currently pressed
   if SpaceKey()=1 then goto MainSection : Rem If the player hits the
spacebar, continue on to game
   REM REFRESH SCREEN
   sync
loop
REM *** END OPTIONS SECTION
REM ***************************************

REM ***************************************
REM *** START MAIN SECTION
REM *** MAIN SECTION HEADER

MainSection:
REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls 0
backdrop on
color backdrop 0
REM OBJECT CREATION
make object cube 1, 100 : Rem Cube
   color object 1,rgb(255,0,0)
   position object 1, 0,50,0
make object plain 2, 5000,5000 : Rem Ground
   color object 2,rgb(255,255,255)
   rotate object 2, 90,0,0
```

```
make object cylinder 3, 100 : Rem X axis
   color object 3,rgb(255,0,0)
   position object 3, 0,0,0
   scale object 3, 5,5000,5
   rotate object 3, 0,0,90
make object cylinder 4, 100 : Rem Y axis
   color object 4,rgb(0,255,0)
   position object 4, 0,0,0
   scale object 4, 5,5000,5
   rotate object 4, 0,90,0
make object cylinder 5, 100 : Rem Z axis
   color object 5,rgb(0,0,255)
   position object 5, 0,0,0
   scale object 5, 5,5000,5
   rotate object 5, 90,0,0
NumberOfCones=20 : NC = NumberOfCones + 10
for x=10 to NC : Rem Cones
   make object cone x, rnd(200)+50
   color object x,rgb(rnd(105)+150,rnd(105)+150,rnd(105)+150)
   do
      NoCenterX = rnd(2000)-1000
      NoCenterZ = rnd(2000)-1000
      if (NoCenterX > -150 and NoCenterX < 150 and NoCenterZ > -150
and NoCenterZ < 150) = 0 then exit
   loop
   position object x, NoCenterX,object size(x)*.75,NoCenterZ
next x
REM LOAD MODELS
REM SET CAMERA
cpZ# = Newzvalue(object position Z(1),object angle Y(1)-180,250)
cpX# = Newxvalue(object position X(1),object angle Y(1)-180,250)
Position camera cpX#,150,cpZ#
Point  camera  object  position  X(1),object  position  Y(1)+50,object
position Z(1)
REM REFRESH SCREEN
sync

REM *** MAIN SECTION LOOP
do
   if scancode()=0 then exit
      ⬆ If no key is currently being pressed (waits for the user to release all keys)
loop
do
   REM SPECIAL EFFECTS
   REM OBJECT ORIENTATIONS
   1pX# = object position X(1)
   1pY# = object position Y(1)
   1pZ# = object position Z(1)
   1aX# = object angle X(1)
   1aY# = object angle Y(1)
   1aZ# = object angle Z(1)
```

```
   REM CAMERA ORIENTATIONS
   cpX# = camera position X()
   cpY# = camera position Y()
   cpZ# = camera position Z()
   caX# = camera angle X()
   caY# = camera angle Y()
   caZ# = camera angle Z()
   REM LIVE SCREEN DISPLAY
   ink 0,rgb(255,255,255) : set text font "times" : set text size 16
: set text to bold : set text opaque
   if myName$ <> "Anonymous"
   remStart
   Checks to make sure myName$ is not "Anonymous", and only prints
the greeting if it's not
   also acceptable (but clumsier):
      if (myName$ = "Anonymous") = 0
   remEnd
      center text 320,425,"'Greetings, "+myName$+"!'"
   endif
   set cursor 0,0
   ink rgb(255,255,255),0 : print "OBJECT 1"
   Rem Print Object position
   ink rgb(255,0,0),0 : print "X position: ";1pX#
   ink rgb(0,255,0),0 : print "Y position: ";1pY#
   ink rgb(0,0,255),0 : print "Z position: ";1pZ#
   print
   Rem Print Object angles
   ink rgb(255,0,0),0 : print "X angle: ";1aX#
   ink rgb(0,255,0),0 : print "Y angle: ";1aY#
   ink rgb(0,0,255),0 : print "Z angle: ";1aZ#

   Rem Print Camera positions
   set cursor 450,0 : ink rgb(255,255,255),0 : print "CAMERA"
   set cursor 450,16 : ink rgb(255,0,0),0 : print "Camera X position:
";cpX#
   set cursor 450,32 : ink rgb(0,255,0),0 : print "Camera Y position:
";cpY#
   set cursor 450,48 : ink rgb(0,0,255),0 : print "Camera Z position:
";cpZ#
   Rem Print Camera angles
   set cursor 450,80 : ink rgb(255,0,0),0 : print "Camera X angle:
";caX#
   set cursor 450,96 : ink rgb(0,255,0),0 : print "Camera Y angle:
";caY#
   set cursor 450,112 : ink rgb(0,0,255),0 : print "Camera Z angle:
";caZ#
```

```
   REM CONTROL INPUT

   if myInput$ = "Keyboard" : Rem Using the Keyboard Commands
      ⬆ Use these keys if "Keyboard" was selected
      ink 0,rgb(255,255,255) : set text font "times" : set text size
16 : set text to bold : set text opaque
      center text 320,440,"USE ARROW KEYS TO MOVE"
      center text 320,455,"PRESS SPACE BAR TO CHANGE COLOR     |
PRESS 'Q' TO QUIT"
      if Upkey()=1 then move object 1,20
         ⬆ If the Up Arrow Key is currently pressed
      if Downkey()=1 then move object 1,-20
         ⬆ If the Down Arrow Key is currently pressed
      if Leftkey()=1 then Yrotate object 1,wrapvalue(1aY#-3.5)
         ⬆ If the Left Arrow Key is currently pressed
      if Rightkey()=1 then Yrotate object 1,wrapvalue(1aY#+3.5)
         ⬆ If the Right Arrow Key is currently pressed
      if Spacekey()=1 : Rem If the player hits the space bar
         ⬆ If the Space Bar is currently pressed
         color object 1,rgb(rnd(255),rnd(255),rnd(255)) : Rem Then
change the color of Object 1
      endif
   endif

   if myInput$ = "Mouse" : Rem Using the Mouse Control
      ⬆ Use these keys if "Mouse" was selected
      ink 0,rgb(255,255,255) : set text font "times" : set text size
16 : set text to bold : set text opaque
      center text 320,440,"USE MOUSE TO MOVE LEFT AND RIGHT   |   USE
BUTTONS TO MOVE BACK AND FORWARD"
      center text 320,455,"PRESS SPACE BAR TO CHANGE COLOR     |
PRESS 'Q' TO QUIT"
      if mouseclick()=1 then move object 1,20
         ⬆ If the Left Mouse Button is currently pressed
      if mouseclick()=2 then move object 1,-20
         ⬆ If the Right Mouse Button is currently pressed
         remStart
         Alternatively, you could use the mouse movement to move the
object back and forward,
        (But it's a little clunky)
            Rem move object 1,0 - (MousemoveY() * 2)
                                  ⬆ Mouse forward and backward movement
         In combination with mouseclick, this could be used for
scrollbars, click and drag, etc.
         remEnd
      Yrotate object 1,wrapvalue(1aY#+MousemoveX()*0.1)
                                  ⬆ Mouse left and right movement
```

```
      if Spacekey()=1 : Rem If the player hits the space bar
```
⬆ If the Space Bar is currently pressed
```
         color object 1,rgb(rnd(255),rnd(255),rnd(255)) : Rem Then
change the color of Object 1
      endif
   endif
```
⬇ Use these keys if “Both” was selected
```
   if myInput$ = "Both" : Rem Using Both the Keyboard and Mouse
Control
      ink 0,rgb(255,255,255) : set text font "times" : set text size
16 : set text to bold : set text opaque
      center text 320,440,"USE MOUSE TO MOVE LEFT AND RIGHT   |   USE
ARROW KEYS TO MOVE BACK AND FORWARD"
      center text 320,455,"CLICK LEFT MOUSE BUTTON TO CHANGE COLOR
|   PRESS 'Q' TO QUIT"
      if Upkey()=1 then move object 1,20
```
⬆ If the Up Arrow Key is currently pressed
```
      if Downkey()=1 then move object 1,-20
```
⬆ If the Down Arrow Key is currently pressed
⬇ If the Left Mouse Button is currently pressed
```
      if mouseclick()=1 then color object 1,
rgb(rnd(255),rnd(255),rnd(255))
      Yrotate object 1,wrapvalue(1aY#+MousemoveX()*0.1)
```
⬆ Mouse left and right movement
```
   endif
```
⬇If the ”q” key is currently pressed
```
   if Inkey$()="q"
      for x = 1 to 100
         if object exist(x) = 1 then delete object x
      next x
      backdrop off
      goto EndSection
   endif
   REM TRANSFORM OBJECTS
   REM MOVE OBJECTS
   REM CHECK FOR COLLISION
   REM MOVE CAMERA
   cpZ# = Newzvalue(1pZ#,1aY#-180,250)
   cpX# = Newxvalue(1pX#,1aY#-180,250)
   Position camera cpX#,150,cpZ#
   Point camera 1pX#,1pY#+50,1pZ#
   REM REFRESH SCREEN
   sync
loop

REM *** STOP MAIN SECTION
REM ************************************
```

```
REM ****************************************
REM *** START END SECTION
REM *** END SECTION HEADER

EndSection:
REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls 0
ink rgb(255,255,255),0 : set text font "times" : set text size 20 :
set text to bold : set text transparent
center text 320,240,"PLAY AGAIN [Y/N]?"
REM LOAD SOUNDS
REM SPECIAL EFFECTS
REM REFRESH SCREEN
sync

REM *** END SECTION LOOP
do
   if scancode()=0 then exit
      ⬆ If no key is currently being pressed (waits for the user to release all keys)
loop
do
   REM CONTROL INPUT
   if Inkey$()="y" then goto OptionsSection
      ⬆If the ”y” key is currently pressed
   if Inkey$()="n"
      ⬆If the ”n” key is currently pressed
      cls : end
   endif
   REM REFRESH SCREEN
   sync
loop

end
REM *** STOP END SECTION
REM ****************************************
```

Collision

What's a video game without some good bumping into things? Think about it. How many games do you play where you collide with other cars, ram into a tree, or simply bump into a bloodthirsty monster (who proceeds to eat you)? In video games, these "bumps" are called "collisions". Collision detection can be one of the most complex parts of developing a game. The reason is that the objects in the 3D game don't actually exist. So determining the *exact boundaries* of an object and what is really a "collision" can be tough. Essentially, collision detection is detecting when collision spaces touch. These spaces can be defined as the objects themselves or as collision boxes in the game engine.

VIDEO: Collisions (4:06)

WORKSHEET

What is a collision?

A collision is when one object's space invades another object's space AND collision detection has been turned on for BOTH objects.

What commands would turn on and off the collision detection for an object with an ID of 2?

SET OBJECT COLLISION ON 2

SET OBJECT COLLISION OFF 2

Your Action:

Load in ***certfication4_2.dba*** and change the collision from *on* to *off* for object 6. If you are not sure how, then go back and watch the Collisions video.

WORKSHEET

In the Collision game certification4_2.dba, when you turned off the collision for object 7, what happened when the cube collided with the cone?

The cube passed through the cone.

certification4_2.dba

```
REM ****************************************
RemStart
    ***
    *** TITLE - Collision
    *** VERSION - 4.2 LAST UPDATED - 1.1.2000
    *** DEVELOPER - Jason Holm
    *** COPYRIGHT - Ingenious Student Labs
    *** DATE CREATED - 1.1.2000
    ***
RemEnd
REM *** START SYSTEM SETUP SECTION

sync on : sync rate 30
autocam off
hide mouse
randomize timer()

REM *** STOP SYSTEM SETUP SECTION
REM ****************************************

REM ****************************************
REM *** START INTRO SECTION
REM *** INTRO SECTION HEADER

REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls 0
load bitmap "images/cubetitle.bmp"
ink rgb(255,255,255),0 : set text font "arial" : set text size 14 :
set text to normal : set text transparent
center text 320,460,"Copyright (c) 2000 Ingenious Student Labs"
REM LOAD SOUNDS
REM SPECIAL EFFECTS
REM REFRESH SCREEN
sync

REM *** INTRO SECTION LOOP

do
   REM SCREEN DISPLAY
   REM CONTROL INPUT
   x=scancode() : if Keystate(x)=1 then goto OptionsSection
   REM REFRESH SCREEN
   sync
loop

REM *** END INTRO SECTION
REM ****************************************
```

```
REM ****************************************
REM *** START OPTIONS SECTION
REM *** OPTIONS SECTION HEADER

OptionsSection:
REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls 0
REM LOAD SOUNDS
REM SPECIAL EFFECTS
REM REFRESH SCREEN
sync

REM *** OPTIONS SECTION LOOP
do
   if scancode()=0 then exit
loop
do
   REM SCREEN DISPLAY
   REM CONTROL INPUT
   goto MainSection
   REM REFRESH SCREEN
   sync
loop
REM *** END OPTIONS SECTION
REM ****************************************

REM ****************************************
REM *** START MAIN SECTION
REM *** MAIN SECTION HEADER

MainSection:
REM DECLARE VARIABLES
energy1 = 0
energy6 = 0
REM SCREEN DISPLAY
cls 0
backdrop on
color backdrop 0
REM OBJECT CREATION
make object cube 1, 100 : Rem Cube
   color object 1,rgb(255,0,0)
   position object 1, 0,50,0
make object plain 2, 2000,2000 : Rem Ground
   color object 2,rgb(255,255,255)
   rotate object 2, 90,0,0
make object cylinder 3, 100 : Rem X axis
   color object 3,rgb(255,0,0)
   position object 3, 0,0,0
   scale object 3, 5,5000,5
   rotate object 3, 0,0,90
```

```
make object cylinder 4, 100 : Rem Y axis
   color object 4,rgb(0,255,0)
   position object 4, 0,0,0
   scale object 4, 5,5000,5
   rotate object 4, 0,0,0
make object cylinder 5, 100 : Rem Z axis
   color object 5,rgb(0,0,255)
   position object 5, 0,0,0
   scale object 5, 5,5000,5
   rotate object 5, 90,0,0
make object cylinder 6, 200 : Rem Cylinder
   color object 6,rgb(0,255,255)
   position object 6, -300,100,500
make object cone 7, 200 : Rem Cone
   color object 7,rgb(255,0,255)
   position object 7, 300,object size(7)*.75,500
REM LOAD MODELS
REM SET COLLISIONS
set object collision on 1 : Rem Check the Player Cube
   make object collision box 1, -50,-50,-50, 50,50,50, 1
   Rem Starting at the lower left corner, we define a box around the
object to check for collision
set object collision off 2 : Rem Don't check the Plain
set object collision off 3 : Rem Don't check the X axis pole
set object collision off 4 : Rem Don't check the Y axis pole
set object collision off 5 : Rem Don't check the Z axis pole
                          ⬇ adjust the collision settings here
set object collision on 6 : Rem Check the left Cylinder
   make object collision box 6, -90,-90,-90, 90,90,90, 1
set object collision on 7 : Rem Check the right Cone
   make object collision box 7, -90,-90,-90, 90,90,90, 1
REM SET CAMERA
cpZ# = Newzvalue(object position Z(1),object angle Y(1)-180,250)
cpX# = Newxvalue(object position X(1),object angle Y(1)-180,250)
Position camera cpX#,150,cpZ#
Point camera object position X(1),object position Y(1)+50,object position Z(1)
REM REFRESH SCREEN
sync

REM *** MAIN SECTION LOOP
do
   if scancode()=0 then exit
loop
do
   REM SPECIAL EFFECTS
   REM OBJECT ORIENTATIONS
   1pX# = object position X(1)
   1pY# = object position Y(1)
   1pZ# = object position Z(1)
   1aX# = object angle X(1)
   1aY# = object angle Y(1)
   1aZ# = object angle Z(1)
```

```
   REM CAMERA ORIENTATIONS
   cpX# = camera position X()
   cpY# = camera position Y()
   cpZ# = camera position Z()
   caX# = camera angle X()
   caY# = camera angle Y()
   caZ# = camera angle Z()
   REM LIVE SCREEN DISPLAY
   ink 0,rgb(255,255,255) : set text font "times" : set text size 16
: set text to bold : set text opaque
   center text 320,420,"TRANSFER ENERGY FROM THE PURPLE CONE TO THE
CYLINDER"
   center text 320,440,"USE ARROW KEYS TO MOVE    |    PRESS 'Q' TO
QUIT"
   set cursor 0,0
   ink rgb(255,255,255),0 : print "OBJECT 1"
   Rem Print Object position
   ink rgb(255,0,0),0 : print "X position: ";1pX#
   ink rgb(0,255,0),0 : print "Y position: ";1pY#
   ink rgb(0,0,255),0 : print "Z position: ";1pZ#
   print
   Rem Print Object angles
   ink rgb(255,0,0),0 : print "X angle: ";1aX#
   ink rgb(0,255,0),0 : print "Y angle: ";1aY#
   ink rgb(0,0,255),0 : print "Z angle: ";1aZ#
   print
   Rem Print "Energy Levels"
   ink rgb(150,150,150),0
   print "MY ENERGY LEVEL: ";energy1
   print "CYLINDER ENERGY STORED: ";energy6
   print
   REM CONTROL INPUT
   if Upkey()=1 then move object 1,20
   if Downkey()=1 then move object 1,-20
   if Leftkey()=1 then Yrotate object 1,wrapvalue(1aY#-3.5)
   if Rightkey()=1 then Yrotate object 1,wrapvalue(1aY#+3.5)
   if Inkey$()="q"
      for x = 1 to 100
         if object exist(x) = 1 then delete object x
      next x
      backdrop off
      goto EndSection
   endif
   REM TRANSFORM OBJECTS
   REM MOVE OBJECTS

   REM CHECK FOR COLLISION
   1pXcol# = object position X(1) : Rem Now that we've moved, set new
collision variables
   1pZcol# = object position Z(1)
```

```
   Rem Collision Type 1: Defined areas
      if 1pXcol#<-950 then 1pXcol#=-950 : Rem Make sure I'm not off
the edge of the plain
      if 1pZcol#<-950 then 1pZcol#=-950
      if 1pXcol#>950 then 1pXcol#=950
      if 1pZcol#>950 then 1pZcol#=950
      position object 1, 1pXcol#,1pY#,1pZcol# : Rem If I am, put me
back
   Rem Collision Type 2: Object collision
      if object collision(1,6) = 1 : Rem If Object 1 collides with
Object 6
         if energy1 > 0
            dec energy1
            inc energy6
            color object 6,
rgb(rnd(105)+150,rnd(105)+150,rnd(105)+150) : Rem Change the color of
the cone
         endif
         position object 1, 1pX#,1pY#,1pZ# : Rem Put Object 1 back
where it was
      endif
      if object collision(1,7) = 1 : Rem If Object 1 collides with
Object 7
         inc energy1
         color object 1,rgb(rnd(105)+150,rnd(105)+150,rnd(105)+150) :
Rem Change the color of the cube
         position object 1, 1pX#,1pY#,1pZ# : Rem Put Object 1 back
where it was
      endif
   Rem Reset Positions
   1pX# = 1pXcol# : Rem Now that we've checked for collision, reset
my variables
   1pZ# = 1pZcol#
   1aY# = object angle Y(1)
   REM MOVE CAMERA
   cpZ# = Newzvalue(1pZ#,1aY#-180,250)
   cpX# = Newxvalue(1pX#,1aY#-180,250)
   if cpX#<-999 then cpX#=-999 : Rem Make sure the camera is not off
the edge
   if cpZ#<-999 then cpZ#=-999
   if cpX#>999 then cpX#=999
   if cpZ#>999 then cpZ#=999
   Position camera cpX#,150,cpZ# : Rem If so, re-adjust
   Point camera 1pX#,1pY#+50,1pZ#
   REM REFRESH SCREEN
   sync
loop

REM *** STOP MAIN SECTION
REM ************************************
```

```
REM ****************************************
REM *** START END SECTION
REM *** END SECTION HEADER

EndSection:
REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls 0
ink rgb(255,255,255),0 : set text font "times" : set text size 20 :
set text to bold : set text transparent
center text 320,240,"PLAY AGAIN [Y/N]?"
REM LOAD SOUNDS
REM SPECIAL EFFECTS
REM REFRESH SCREEN
sync

REM *** END SECTION LOOP
do
   if scancode()=0 then exit
loop
do
   REM CONTROL INPUT
   if Inkey$()="y" then goto OptionsSection
   if Inkey$()="n"
      cls : end
   endif
   REM REFRESH SCREEN
   sync
loop

end
REM *** STOP END SECTION
REM ****************************************
```

Textures

Textures are images that are applied to objects. Textures are used to give the objects in a game a realistic look. Basically, textures are used to fool the eye into seeing 3D detail that doesn't really exist. Think about a brick wall. If you stand outside and look at a brick wall, your eyes see the detail of how the bricks are stacked and mortared together. Because of the way the light is hitting the brick surface, your eyes (and brain) process and recognize that the brick surface isn't smooth but is rough and uneven in "texture". Now, take a picture of that wall you have been looking at, run inside and apply that picture to an object in a game you want to look like a wall, and your eye will see a surface that isn't perfectly flat and smooth, but is rough, uneven and has texture. The objects you create in the game engine are created as perfectly smooth 3-dimensional objects. It is the texture that the game designer applies to that object that makes it look as real as a brick wall you might see outside. There are people who make their living in the game industry who only make textures.

The other part of creating textures is that the size and shape are very important.

WORKSHEET
Why would you want to use textures?
Textures can make an object look more realistic.

VIDEO: Textures (3:10)

WORKSHEET
What is the type of image file that can be used as a texture? *.bmp files only*

Your Action:
Start up the Dark Basic editor and take a moment to look through the pre-made models and their textures (.bmp files).

Now to find out how textures are applied to objects in the code watch the following video:

VIDEO: Texture Examples in Game Code (3:36)

WORKSHEET
Where do you find the pre-made models and their textures in Dark Basic?
Under the menu selection MEDIA then under MEDIA BROWSER

Describe a model that had more than one texture and the textures it used.
Any number of models under the MEDIA MEDIA BROWSER use multiple textures. One example is the Penguin that has 3 textures -- body, leg and eye.

Your Action:
Load in ***certification4_3.dba***. First, look in the Main Section to see the different images/textures that are being loaded. Then run the game and identify the different textures applied to the different objects.

certification4_3.dba

```
REM ****************************************
RemStart
    ***
    *** TITLE - Bitmaps and Image Textures
    *** VERSION - 4.3 LAST UPDATED - 1.1.2000
    *** DEVELOPER - Jason Holm
    *** COPYRIGHT - Ingenious Student Labs
    *** DATE CREATED - 1.1.2000
    ***
RemEnd
REM *** START SYSTEM SETUP SECTION

sync on : sync rate 30
autocam off
hide mouse
randomize timer()

REM *** STOP SYSTEM SETUP SECTION
REM ****************************************

REM ****************************************
REM *** START INTRO SECTION
REM *** INTRO SECTION HEADER

REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls 0
load bitmap "images/cubetitle.bmp" : Rem load a still image & display
it
ink rgb(255,255,255),0 : set text font "arial" : set text size 14 :
set text to normal : set text transparent
center text 320,460,"Copyright (c) 2000 Ingenious Student Labs"
REM LOAD SOUNDS
REM SPECIAL EFFECTS
REM REFRESH SCREEN
sync

REM *** INTRO SECTION LOOP

do
   REM SCREEN DISPLAY
   REM CONTROL INPUT
   x=scancode() : if Keystate(x)=1 then goto OptionsSection
   REM REFRESH SCREEN
   sync
loop

REM *** END INTRO SECTION
REM ****************************************
```

```
REM ****************************************
REM *** START OPTIONS SECTION
REM *** OPTIONS SECTION HEADER

OptionsSection:
REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls 0
REM LOAD SOUNDS
REM SPECIAL EFFECTS
REM REFRESH SCREEN
sync

REM *** OPTIONS SECTION LOOP
do
   if scancode()=0 then exit
loop
do
   REM SCREEN DISPLAY
   REM CONTROL INPUT
   goto MainSection
   REM REFRESH SCREEN
   sync
loop
REM *** END OPTIONS SECTION
REM ****************************************

REM ****************************************
REM *** START MAIN SECTION
REM *** MAIN SECTION HEADER

MainSection:
REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls 0
backdrop on
color backdrop 0
REM LOAD IMAGES ⬅ Textures are loaded here
Load image "images/clock.bmp",1 : Rem Import images and assign ids to
them
Load image "images/grassy01.bmp",2
Load image "images/rock12.bmp",3
Load image "images/rocky05.bmp",4
texture backdrop 4 : Rem Apply texture 4 to the backdrop
REM LOAD SOUNDS
```

```
REM OBJECT CREATION ⬅Textures are applied as the objects are created
make object cube 1, 100 : Rem Cube
   Texture object 1,1 : Rem Apply texture 1 to the cube
   scale object 1,100,50,100
   position object 1, 0,25,0
make object plain 2, 2000,2000 : Rem Ground
   Texture object 2,2 : Rem Apply texture 2 to the ground
   scale object texture 2,20,20 : Rem Shrink the texture 20 times
each way so it tiles on the surface
   rotate object 2, 90,0,0
   position object 2, 0,0,0

NumberOfCones=90 : NC = NumberOfCones + 10
for x=10 to NC : Rem Cones
   make object cone x, rnd(200)+50
   Texture object x,3 : Rem Apply texture 3 to each cone
   do
      NoCenterX = rnd(2000)-1000
      NoCenterZ = rnd(2000)-1000
      if (NoCenterX > -150 and NoCenterX < 150 and NoCenterZ > -150
and NoCenterZ < 150) = 0 then exit
   loop
   position object x, NoCenterX,object size(x)*.75,NoCenterZ
next x

REM LOAD MODELS
REM SET COLLISIONS
set object collision on 1 : make object collision box 1, -50,-25,-50,
50,25,50, 1
set object collision off 2

for x=10 to NC
set object collision on x : make object collision box x, 0-(object
size X(x)/3),0-(object size Y(x)/3),0-(object size Z(x)/3), (object
size X(x)/3),(object size Y(x)/3),(object size Z(x)/3), 1
next x

REM SET CAMERA
cpZ# = Newzvalue(object position Z(1),object angle Y(1)-180,250)
cpX# = Newxvalue(object position X(1),object angle Y(1)-180,250)
Position camera cpX#,150,cpZ#
Point camera object position X(1),object position Y(1)+50,object
position Z(1)
REM SPECIAL EFFECTS
REM REFRESH SCREEN
sync
```

```
REM *** MAIN SECTION LOOP
do
   if scancode()=0 then exit
loop
do
   REM SPECIAL EFFECTS
   for x=10 to NC step 3
      Scroll object texture x,0,0.005 : Rem Scroll texture up along
object every time the main section loops
   next x
   for x=10 to NC step 4
      Scroll object texture x,0.005,0 : Rem Scroll texture clockwise
around object every time the main section loops
   next x
   REM OBJECT ORIENTATIONS
   1pX# = object position X(1)
   1pY# = object position Y(1)
   1pZ# = object position Z(1)
   1aX# = object angle X(1)
   1aY# = object angle Y(1)
   1aZ# = object angle Z(1)
   REM CAMERA ORIENTATIONS
   cpX# = camera position X()
   cpY# = camera position Y()
   cpZ# = camera position Z()
   caX# = camera angle X()
   caY# = camera angle Y()
   caZ# = camera angle Z()
   REM LIVE SCREEN DISPLAY
   ink 0,rgb(255,255,255) : set text font "times" : set text size 16
: set text to bold : set text opaque
   center text 320,440,"USE ARROW KEYS TO MOVE    |    PRESS 'Q' TO
QUIT"
   REM CONTROL INPUT
   if Upkey()=1
      move object 1,10
      Scroll object texture 1,0,-0.05 : Rem Scroll texture up and
forward along the cube each time the user hits up
   endif
   if Downkey()=1
      move object 1,-10
      Scroll object texture 1,0,0.05 : Rem Scroll texture down and
backward along the cube each time the user hits down
   endif
   if Leftkey()=1
      Yrotate object 1,wrapvalue(1aY#-3.5)
      backscroll = backscroll-10 : Rem Every time the user hits left,
subtract 10 from the backscroll value
      SCROLL BACKDROP backscroll, 0 : Rem Move the background left or
right, according to the backscroll value
   endif
```

```
   if Rightkey()=1
      Yrotate object 1,wrapvalue(1aY#+3.5)
      backscroll = backscroll+10 : Rem Every time the user hits left,
add 10 from the backscroll value
      SCROLL BACKDROP backscroll, 0 : Rem Move the background left or
right, according to the backscroll value
   endif
   if Inkey$()="q"
      for x = 1 to 100
         if object exist(x) = 1 then delete object x
      next x
      backdrop off
      goto EndSection
   endif
   REM TRANSFORM OBJECTS
   REM MOVE OBJECTS
   REM CHECK FOR COLLISION
   1pXcol# = object position X(1)
   1pZcol# = object position Z(1)
   if 1pXcol#<-950 then 1pXcol#=-950
   if 1pZcol#<-950 then 1pZcol#=-950
   if 1pXcol#>950 then 1pXcol#=950
   if 1pZcol#>950 then 1pZcol#=950
   position object 1, 1pXcol#,1pY#,1pZcol#

   for x=10 to NC
   if object collision(1,x) = 1
       position object 1, 1pX#,1pY#,1pZ#
   endif
   next x

   1pX# = 1pXcol#
   1pZ# = 1pZcol#
   1aY# = object angle Y(1)
   REM MOVE CAMERA
   cpZ# = Newzvalue(1pZ#,1aY#-180,250)
   cpX# = Newxvalue(1pX#,1aY#-180,250)
   if cpX#<-999 then cpX#=-999 : Rem Make sure the camera is not off
the edge
   if cpZ#<-999 then cpZ#=-999
   if cpX#>999 then cpX#=999
   if cpZ#>999 then cpZ#=999
   Position camera cpX#,150,cpZ# : Rem If so, re-adjust
   Point camera 1pX#,1pY#+50,1pZ#
   REM REFRESH SCREEN
   sync
loop

REM *** STOP MAIN SECTION
REM ************************************
```

```
REM ****************************************
REM *** START END SECTION
REM *** END SECTION HEADER

EndSection:
REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls 0
ink rgb(255,255,255),0 : set text font "times" : set text size 20 :
set text to bold : set text transparent
center text 320,240,"PLAY AGAIN [Y/N]?"
REM LOAD SOUNDS
REM SPECIAL EFFECTS
REM REFRESH SCREEN
sync

REM *** END SECTION LOOP
do
   if scancode()=0 then exit
loop
do
   REM CONTROL INPUT
   if Inkey$()="y" then goto OptionsSection
   if Inkey$()="n"
      cls : end
   endif
   REM REFRESH SCREEN
   sync
loop

end
REM *** STOP END SECTION
REM ****************************************
```

Sounds

Sounds add a great deal to a video game. Adding sound -- *good* sound -- to your game is important if you expect your game to be well received. Dull, boring sounds, slow music, poor sound quality, or no sound at all are all fast ways to get a player to turn off your game. Fortunately, it is very easy to add sound -- even your own voice -- to a game.

Watch the following video to learn about how sound is imported and used inside a game program.

VIDEO: Sounds in Game Code (4:18)

WORKSHEET

What type of sound files can you use with this game engine?
.wav and .mp3 files

What is the command to load in a new sound?
Load sound "sounds/scifi5.wav",1

What command will play a sound once?
Play sound 1

What command will play a sound continuously?
Loop sound 7

Your Action:

Load in ***certification4_4.dba***. Change the volume of the song by going to the Main Section Header and finding the line of code below REM SOUND EFFECTS.

set sound volume 7,50 : Rem Lower the volume of sound 7

Change the 50 to 75. Run the program. Now see what happens when you change the volume to 25.

WORKSHEET

What command do you use to set the volume of sound 7 to a level of 50?
Set sound volume 7,50

certification4_4.dba

```
REM ****************************************
RemStart
    ***
    *** TITLE - Sounds
    *** VERSION - 4.4 LAST UPDATED - 1.1.2000
    *** DEVELOPER - Jason Holm
    *** COPYRIGHT - Ingenious Student Labs
    *** DATE CREATED - 1.1.2000
    ***
RemEnd
REM *** START SYSTEM SETUP SECTION

sync on : sync rate 30
autocam off
hide mouse
randomize timer()

REM *** STOP SYSTEM SETUP SECTION
REM ****************************************

REM ****************************************
REM *** START INTRO SECTION
REM *** INTRO SECTION HEADER

REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls 0
load bitmap "images/cubetitle.bmp"
ink rgb(255,255,255),0 : set text font "arial" : set text size 14 :
set text to normal : set text transparent
center text 320,460,"Copyright (c) 2000 Ingenious Student Labs"
center text 320,440,"Music courtesy of LipRiddle"
REM LOAD SOUNDS
REM SPECIAL EFFECTS
REM REFRESH SCREEN
sync

REM *** INTRO SECTION LOOP

do
   REM SCREEN DISPLAY
   REM CONTROL INPUT
   x=scancode() : if Keystate(x)=1 then goto OptionsSection
   REM REFRESH SCREEN
   sync
loop

REM *** END INTRO SECTION
REM ****************************************
```

```
REM ****************************************
REM *** START OPTIONS SECTION
REM *** OPTIONS SECTION HEADER

OptionsSection:
REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls 0
ink rgb(255,255,255),0 : set text font "times" : set text size 20 :
set text to bold : set text transparent
center text 320,240,"LOADING SOUNDS" : Rem Let the player know why
the game isn't starting up immediately
sync
wait 1
REM LOAD SOUNDS ⬅ Sounds are loaded here
load sound "sounds/scifi5.wav",1 : Rem Load each sound file and give
it an id number
load sound "sounds/scifi4.wav",2
load sound "sounds/powerhum.wav",3
load sound "sounds/walk.wav",4
load sound "sounds/pop.wav",5
load sound "sounds/thunder.wav",6
load sound "sounds/wrongway.mp3",7
REM SPECIAL EFFECTS
REM REFRESH SCREEN

REM *** OPTIONS SECTION LOOP
do
   if scancode()=0 then exit
loop
do
   REM SCREEN DISPLAY
   REM CONTROL INPUT
   goto MainSection
   REM REFRESH SCREEN
   sync
loop
REM *** END OPTIONS SECTION
REM ****************************************

REM ****************************************
REM *** START MAIN SECTION
REM *** MAIN SECTION HEADER

MainSection:
REM DECLARE VARIABLES
energy1 = 0
energy6 = 0
REM SCREEN DISPLAY
cls 0
backdrop on
color backdrop 0
```

```
REM SOUND EFFECTS ⬅ Sounds are activated and edited here
play sound 1 : Rem Play a sound once
loop sound 7 : Rem Play a sound continuously
set sound volume 7,50 : Rem Lower the volume of sound 7
REM OBJECT CREATION
make object cube 1, 100 : Rem Cube
   color object 1,rgb(255,0,0)
   position object 1, 0,50,0
make object plain 2, 2000,2000 : Rem Ground
   color object 2,rgb(255,255,255)
   rotate object 2, 90,0,0
make object cylinder 3, 100 : Rem X axis
   color object 3,rgb(255,0,0)
   position object 3, 0,0,0
   scale object 3, 5,5000,5
   rotate object 3, 0,0,90
make object cylinder 4, 100 : Rem Y axis
   color object 4,rgb(0,255,0)
   position object 4, 0,0,0
   scale object 4, 5,5000,5
   rotate object 4, 0,0,0
make object cylinder 5, 100 : Rem Z axis
   color object 5,rgb(0,0,255)
   position object 5, 0,0,0
   scale object 5, 5,5000,5
   rotate object 5, 90,0,0
make object cylinder 6, 200 : Rem Cylinder
   color object 6,rgb(0,255,255)
   position object 6, -300,100,500
make object cone 7, 200 : Rem Cone
   color object 7,rgb(255,0,255)
   position object 7, 300,object size(7)*.75,500
REM LOAD MODELS
REM SET COLLISIONS
set object collision on 1
   make object collision box 1, -50,-50,-50, 50,50,50, 1
set object collision off 2
set object collision off 3
set object collision off 4
set object collision off 5
set object collision on 6
   make object collision box 6, -90,-90,-90, 90,90,90, 1
set object collision on 7
   make object collision box 7, -90,-90,-90, 90,90,90, 1
REM SET CAMERA
cpZ# = Newzvalue(object position Z(1),object angle Y(1)-180,250)
cpX# = Newxvalue(object position X(1),object angle Y(1)-180,250)
Position camera cpX#,150,cpZ#
Point  camera  object  position  X(1),object  position  Y(1)+50,object
position Z(1)
REM REFRESH SCREEN
sync
```

```
REM *** MAIN SECTION LOOP
do
   if scancode()=0 then exit
loop
do
   REM SPECIAL EFFECTS
   REM OBJECT ORIENTATIONS
   1pX# = object position X(1)
   1pY# = object position Y(1)
   1pZ# = object position Z(1)
   1aX# = object angle X(1)
   1aY# = object angle Y(1)
   1aZ# = object angle Z(1)
   REM CAMERA ORIENTATIONS
   cpX# = camera position X()
   cpY# = camera position Y()
   cpZ# = camera position Z()
   caX# = camera angle X()
   caY# = camera angle Y()
   caZ# = camera angle Z()
   REM LIVE SCREEN DISPLAY
   ink 0,rgb(255,255,255) : set text font "times" : set text size 16
: set text to bold : set text opaque
   center text 320,432,"TRANSFER ENERGY FROM THE PURPLE CONE TO THE
CYLINDER"
   center text 320,448,"PRESS 'P' TO PAUSE THE MUSIC    |    PRESS
SPACE BAR TO PLAY SOUND 5"
   center text 320,464,"USE ARROW KEYS TO MOVE    |    PRESS 'Q' TO
QUIT"
   set cursor 0,0
   ink rgb(255,255,255),0 : print "OBJECT 1"
   Rem Print Object position
   ink rgb(255,0,0),0 : print "X position: ";1pX#
   ink rgb(0,255,0),0 : print "Y position: ";1pY#
   ink rgb(0,0,255),0 : print "Z position: ";1pZ#
   print
   Rem Print Object angles
   ink rgb(255,0,0),0 : print "X angle: ";1aX#
   ink rgb(0,255,0),0 : print "Y angle: ";1aY#
   ink rgb(0,0,255),0 : print "Z angle: ";1aZ#
   print
   Rem Print "Energy Levels"
   ink rgb(150,150,150),0
   print "MY ENERGY LEVEL: ";energy1
   print "CYLINDER ENERGY STORED: ";energy6
   print
   print "MUSIC VOLUME: ";get sound volume(7) : Rem Display the
volume of sound 7
```

```
   REM CONTROL INPUT
   if Upkey()=1
      move object 1,20
      if sound playing(4)=0 then play sound 4 : Rem If sound 4 isn't
already playing, then start it
   endif

   if Downkey()=1
      move object 1,-20
      if sound playing(4)=0 then play sound 4 : Rem If sound 4 isn't
already playing, then start it
   endif

   if Leftkey()=1 then Yrotate object 1,wrapvalue(1aY#-3.5)
   if Rightkey()=1 then Yrotate object 1,wrapvalue(1aY#+3.5)

   if Inkey$()="p" : Rem If the player hits 'p'
      if hitp = 0 : Rem and if the player has let go from before
         if sound paused(7)=1 then resume sound 7 else pause sound 7
            Rem If the sound is paused, resume it. If it's resumed,
pause it
         hitp = 1 : Rem Record that the player hasn't let go of the
'p' key yet
      endif
   else : Rem If the player is not hitting the 'p' key
      hitp = 0 : Rem Record that the player has let go of the 'p' key
   endif

   if Spacekey()=1 then play sound 5 : Rem start playing sound 5
every time the spacebar is hit

   if Inkey$()="q"
      for x = 1 to 100
         if object exist(x) = 1 then delete object x
      next x
      backdrop off
      goto EndSection
   endif
   REM TRANSFORM OBJECTS
   REM MOVE OBJECTS
   REM CHECK FOR COLLISION
   1pXcol# = object position X(1)
   1pZcol# = object position Z(1)
   Rem Arena Collision
      if 1pXcol#<-950 then 1pXcol#=-950
      if 1pZcol#<-950 then 1pZcol#=-950
      if 1pXcol#>950 then 1pXcol#=950
      if 1pZcol#>950 then 1pZcol#=950
      position object 1, 1pXcol#,1pY#,1pZcol#
```

```
   Rem Storage cylinder collision
      if object collision(1,6) = 1
         if energy1 > 0
            dec energy1
            inc energy6
            color object 6,
rgb(rnd(105)+150,rnd(105)+150,rnd(105)+150)
            if sound playing(2)=0 then play sound 2 : Rem If sound 2
isn't already playing, then start it
         else
            stop sound 2 : Rem If energy1=0 then stop playing sound 2
         endif
         position object 1, 1pX#,1pY#,1pZ#
      else
         if sound playing(2)=1 then stop sound 2
            Rem If the cube isn't colliding with the cylinder and
sound 2 is still playing, stop it
      endif
   Rem Source Cone Collision
      if object collision(1,7) = 1
         inc energy1
         color object 1,rgb(rnd(105)+150,rnd(105)+150,rnd(105)+150)
         if sound playing(3)=0 then play sound 3 : Rem If sound 3
isn't already playing, then start it
         position object 1, 1pX#,1pY#,1pZ#
      else
         if sound playing(3)=1 then stop sound 3
            Rem If the cube isn't colliding with the cone and sound 3
is still playing, stop it
      endif

   Rem Reset Positions
   1pX# = 1pXcol#
   1pZ# = 1pZcol#
   1aY# = object angle Y(1)
   REM MOVE CAMERA
   cpZ# = Newzvalue(1pZ#,1aY#-180,300)
   cpX# = Newxvalue(1pX#,1aY#-180,300)
   if cpX#<-999 then cpX#=-999
   if cpZ#<-999 then cpZ#=-999
   if cpX#>999 then cpX#=999
   if cpZ#>999 then cpZ#=999
   Position camera cpX#,150,cpZ#
   Point camera 1pX#,1pY#+50,1pZ#
   REM REFRESH SCREEN
   sync
loop

REM *** STOP MAIN SECTION
REM ************************************
```

```
REM ****************************************
REM *** START END SECTION
REM *** END SECTION HEADER

EndSection:
REM DECLARE VARIABLES
REM SCREEN DISPLAY
cls 0
ink rgb(255,255,255),0 : set text font "times" : set text size 20 :
set text to bold : set text transparent
center text 320,240,"PLAY AGAIN [Y/N]?"
REM SOUND EFFECTS
for x = 1 to 1024
   if sound exist(x) =1 then stop sound x
next x
play sound 6
REM SPECIAL EFFECTS
REM REFRESH SCREEN
sync

REM *** END SECTION LOOP
do
   if scancode()=0 then exit
loop
do
   REM CONTROL INPUT
   if Inkey$()="y" then goto OptionsSection
   if Inkey$()="n"
      cls : end
   endif
   REM REFRESH SCREEN
   sync
loop

end
REM *** STOP END SECTION
REM ****************************************
```

Packing and Final EXE

A very important part of game design is packing and making a final EXE. This means that the game is done and ready to share with the world. It also means that you might actually get the opportunity to sell your game. Nobody wants to buy a game that is only partially finished. You can create a "beta" of your game to share. A *beta* is a piece of software that is still in testing and that the developer takes no liability on whether or not the software functions completely. Typically, game companies will let a few betas out for key people to test, but for the most part, you never want to distribute any program you have not fully tested.

It is very simple to create a final EXE, or executable, file. The EXE is the file that you can save onto a CD and then share with your friends or family. To find out how to create a final EXE of your game, watch the following video.

VIDEO: Making a Final EXE (1:22)

WORKSHEET

What is the purpose of creating a final EXE?

It allows you to distribute the game.

Final Packaging Checklist

_____ The game is in its own main directory with a unique name.

_____ All the files and resources needed for the game are in the proper sub-directories under the game's own main directory.

_____ All paths in the main game code correctly point to the different resources.

_____ Only resources necessary for the game are included in the sub-directories.

Organization of the Game Directory

To keep the game from being "bulked up", place only the files used in the game inside the sub-directories. Also, the only file that should be in the main directory is the main game program.

The game's files should be contained in a file structure that looks like this:

MyGame	(Main Directory)
anims	(Sub-directory for animations and videos)
images	(Sub-directory for images)
models	(Sub-directory for 3D models)
sounds	(Sub-directory for sounds)
music	(Sub-directory for music)

Moving a game to a new folder before creating a FINAL package

If you are moving your main program and its parts to a new directory, you must make sure that all of the references for the different resources (images, sounds, models, animations) have correct paths in the code. For example, the correct path for any of the sub-directories shown above is the **directory name/name of file including extension**. An image named *can.bmp* has a path of **images/can.bmp**.

Error 10

If you have a path incorrect, or your final packaged game cannot find a file, you will see the Runtime Error 10 box appear. If you see this when running your final packaged game, go back and check to see that all the resources are in their correct sub-directories and that the paths in the main program correctly point to the resource.

Additional Resources

9-Weeks Plan	Project	Phase
1	One / Carbonade	1-2
2	One / Carbonade	3
3	One / Carbonade	4
4	One / Carbonade	4-5
5	One / Carbonade	5
6	One / Carbonade	5
7	One / Carbonade	5
8	One / Carbonade	6
9	One / Carbonade	7

18-Weeks Plan	Project	Phase
1	One / Carbonade	1-2
2	One / Carbonade	3
3	One / Carbonade	4
4	One / Carbonade	4-5
5	One / Carbonade	5
6	One / Carbonade	5
7	One / Carbonade	6
8	One / Carbonade	7
9	Wrap-Up	Portfolio & Finalize .exe
10	Two / Bacteria Bash	1
11	Two / Bacteria Bash	2
12	Two / Bacteria Bash	3
13	Two / Bacteria Bash	4
14	Two / Bacteria Bash	5
15	Two / Bacteria Bash	5
16	Two / Bacteria Bash	6
17	Two / Bacteria Bash	6-7
18	Two / Bacteria Bash	Wrap-Up

36-Weeks Plan	Project	Phase
Weeks 1-6	One / Carbonade	1-3
Weeks 7-12	One / Carbonade	4-5
Weeks 13-18	One / Carbonade	6-7
Weeks 19-24	Two / Bacteria Bash	1-3
Weeks 25-30	Two / Bacteria Bash	4-5
Weeks 31-36	Two / Bacteria Bash	6-7

Goals for the Development Meeting (Initial Staff Meeting)

Purpose

- To foster the environment and to activate prior knowledge, as well as be part of the anticipatory set of the class

Environment(s) to Foster/Establish

- Culture (Home)
- Business Office
- Teacher/Supervisor—Student/Employee

Teacher Benefits

- What experiences they have had w/ video games
- What programming experiences they have had
- Become aware of "personal situations" like culture, home-life, etc.

Student Gains

- Expectations
- Further explain the program structure
- A connection that would allow the student to feel comfortable asking questions

Script—Initial Meeting (Whole Class)

"This class is an opportunity to create video games and will be a different setting than the traditional classroom setting."

"You will be interns working for Ingenious Student Labs, where you will be meeting the expectations of a company, learning job skills and professional behavior."

"My job will have multiple hats including being the teacher but also being your supervisor, and often times a consultant."

"You will be expected to meet certain training requirements including becoming certified in Dark Basic. To start with, I would like you to watch this video that will introduce you to the use of the software. (Show student orientation video.) I want you to know that anytime that you have a question for me as a supervisor/teacher you can email me through the system using the PDA."

"Knowing that you will be designing a video game, I want you to begin thinking about the types of video games that you currently play." (Begin making a concept web.)

[1st Person Shooter, RTS (Real-Time Strategy), RPG (Role Playing Game), Sports, Platforming, Puzzles]

"What specific games do you like to play regarding these types of video games?"
[1st Person Shooter—Halo; RTS—Warcraft; RPG—SIMS; Sports—Madden NFL; Platforming—SuperMario Bros.; Puzzle--Tetris
"What are specific features of these games that you enjoy in the game? What sets it apart from other similar games?"

Relate with the students how the games that you have been discussing relates to what they could program their video game to do.

Follow-up Development Meetings

Development Meeting (Post Phase 4??) Small Group Meeting??

- Discuss with the students how they can plan and develop ideas for video games using a flowchart organizational system and a storyboard. You could use one of the example video games from the student files to model this.

Development Meeting (Check for Understanding)--Can be used anytime with Small Group or Whole Class

- Use the KWL find out what the students have learned thus far and what interests the students have developed with the curriculum. This is an excellent opportunity to review essential vocabulary and concepts that the students have been working with in the curriculum.

Development Meeting (Showcase student learning and discovery)--Whole Class

- This meeting can be just a short meeting to celebrate student learning and discovery in your classroom. *Or* it could be added on to another planned meeting. For example, if *Susie* found a particular color or texture that is unusual have her share with the class how she found it.

Development Meeting (Review and informally assess)

- Use essential vocabulary words and concepts to play a game similar to Pictionary with the students. This is fun way to review terms and concepts that students use in the curriculum.

3-D Coordinate System/Starship Game

Objective: The student will practice using a 3-D coordinate system (x, y, z) by finding locations in a room setting.

Materials: Various objects (balls, stuffed animals, chairs, desks, etc.); red yarn; green yarn; blue yarn; sticky notes/notecards; masking tape; 2 computers with IM capabilities or 2 walkie-talkies (optional for game)

Set-up: Set up classroom to be a 3-D Coordinate System with red yarn suspended from the walls left to right for the x-axis, green yarn suspended from the middle of the ceiling to the middle of the floor for a y-axis, and blue yarn suspended from the walls from back to front for a z-axis. The center of the room where all 3 strings meet is (0, 0, 0). Label the axes at points such as 0, 1000, 2000, 3000, etc.

Anticipatory Set: Involve students in a conversation regarding the game of Battleship, include conversation points that involve the x- and y-axes. Allude to the students that they will be participating in a game similar to Battleship that also involves the z-axis.

Teacher Input/Modeling:

1. Review the x-, y- and z-axes with students and locate various objects in the room. For example: desks, chairs, etc. Students may be asked to give coordinate locations of other objects to check for understanding.
2. Students can also be involved in this as well (keep their attention!) by the following set of instructions:
   - "If I have a desk located at (1000, -3000, 4000), and I wanted to place *Billy* behind the desk according to our current camera view, what coordinates would I place *Billy* at?"
   -"If I have a chair located at (2000, -2500, 3300), and I wanted to place *Susie* on top of the chair, what coordinates would I place *Susie* at?"
   -Hold various objects up in the room and have students state their locations.

Guided Practice: Have the students find a place in the room to sit, stand, etc. and then ask each one to give their coordinate location. Assist as needed.

Guided/Independent Practice: (Optional Game)

1. Set up a 3-D coordinate system in two adjoining rooms.
2. Divide students into two teams.
3. Have students place 1 or 2 objects in various parts of the coordinate grid. (Objects may need some "help" with elevation on the y-axis, such as placing a stuffed bear on top of a chair.)
4. Using computers with Instant Messaging capabilities or walkie-talkies, have Team 1 fire at a location in the opposite room. Members from Team 2 would then indicate to the other students where the shot was located in relation to the object (to the left or right, above or below, in front of or in back of).
5. Team 2 would then fire at Team 1's object and so on.

Closure: After bringing students back together, discuss what they learned from the experience. Discuss how this relates to placing objects in a video game. Illustrate this by running the program Certification 3_4 in DarkBASIC.

Lesson created by: Kendal Berrey
I Support Learning, Inc.

Ideas for Classroom

- Following the internship emphasis, have students create business cards for themselves
- When sending emails to interns, sign off as their supervisor
- Create a company name for your classroom
- Show "The Making of Finding Nemo" (documentary section of Disk 1)
    - To emphasize teamwork; storyboarding; importance of planning; demonstrate thought process for large projects
- After Bacteria Bash, have students return to the game they created in Carbonade and apply the skills they learned in Project 2
- If your district has access to United Streaming, use the video of the interview with Nathan Sitkoff, video game designer
- Cheat Sheet: have students write shortcuts in a central location, then during "development meetings" have them give details on the shortcut.
- Have students create commercial after completing their game
- Have students design the cover for their game