Game Builders Academy™

Learn ★ Grow ★ Have Fun ★ Succeed!

Video Game Design and Development

STARBURST Grades K-4

Michael Pugliese
Sally Rosenberg
Phil Lipsky

TEAM GBA CORP. 35 Lace Lane, Westbury, New York 11590

GAME BUILDERS ACADEMY™

Learn ★ Grow ★ Have Fun ★ Succeed!

STARBURST
Video Game Design and Development
written by Michael Pugliese

Technical Editors/Consultants
Stacey Rosenberg
Temma Clark-Braverman

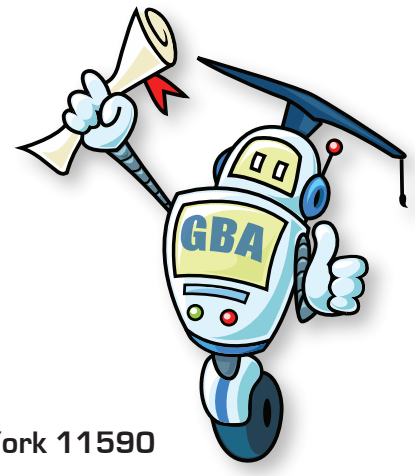
Production Editors
Sally Brecher Rosenberg
Phil Lipsky

Box/Binder Design
Michael Pugliese

Game Art
Michael Pugliese

Marketing and Sales
Walter Ebe
John Taylor

ISBN-13:
978-0-9815502-2-0



Welcome to this Exciting Teaching Tool

Enjoy this wonderful journey with your students - and let us know about your successes!

Course Overview

This course is designed to guide you and your students through the process of building a full, working video game. Along the way, there are many opportunities for students to apply, practice, and reinforce various academic subjects they've learned or are learning in school.

In this course, your students will use and strengthen their math skills, logic skills, communication skills, concentration and critical thinking skills, problem solving and creative thinking – all in the context of learning how to design and program their own video game. Your students will be introduced to the technical and artistic concepts and techniques of designing and building a video game. Students will create some of their own art for their games. Students will also be introduced to the fundamentals of animation for use in their games.

Students will use the free version of Game Maker software to create their games. Game Maker is a software (program) that enables the user to create video games in either a full click-button (click-and-drag) environment, or fully by writing programming code, or using a combination of both. In this Starburst Level 1 course, all game development will be done using the click-button method.

How to Use this Manual

This manual is designed to guide you and your students through the process of building a full, working video game. Along the way there will be many opportunities for discussion about various academic subjects, creative and artistic ideas, and much more.

The project is divided into 10 lessons. Each lesson consists of a number of modules. The modules are designed to be, as the name implies, modular. Many of them can be done in different sequence or on their own for variations on the given project. But we highly recommend that you follow the sequence of lessons and modules in as laid out in this manual. The order of modules has been carefully designed to build sequentially to the finished project, and to build students' knowledge, skills, and confidence with each successive module.

Each module includes the detailed procedure(s), in a clear, numbered, step-by-step organization. Indented and offset within each module are questions to ask students (along with answers), additional explanations and clarifications on given steps, ideas for having students reinforce material or experiment with skills newly gained in the given module (or step), and more. We highly recommend that you incorporate as much of this material as possible.

Throughout the manual, there are also many sidebars on the right-hand side of the pages. These contain additional information, ideas for variations on the given module, ideas for discussion, academic connections, important information, and more. As with the indented material in the body of the modules, we recommend that you include as much of the sidebar material as possible for your class.

Show the sample game (Jumper Quest) to your students during the first day of class so they can see the end goal, and so they will have a clear picture and overview of the project.

Along with the finished sample game, there is a file that accompanies each module and shows how the game should look up to, and after, completion of the module. These are called "postmods" for post module, or "after" module. It's usually helpful to show the postmod for a given module before teaching that module, but this is up to you. You may choose to have your students fully discover the result of their work on their own, without first seeing what the result will be.

If a student misses one or more classes, when they return, they can use the Module start file located on the Instructor Files as their "starter" file on the day they return. Although this is, of course, not as good as having the student's own game to work on that day, it at least allows the student to follow, complete, and learn the day's material, with a file that includes all the material that was covered on the day(s) missed.

The postmods also serve as reference for you and for your students as to exactly how the file is set up. So, as a teacher, you have each module on paper, with the steps - and you also have the specific postmod file that goes with that module to show exactly how the file looks with all the steps in the module completed.

Take the opportunities for academic discussions as far and as deep as you like, or not at all, as you see best for your class and the given day, students' responses, classroom situation, etc. **Enjoy this wonderful journey with your students - and let us know about your teaching successes and students' accomplishments!**

What's on the Included Disks

INSTRUCTOR FILES DISK

Link to Game Builders Academy Web site: http://www.gbalearning.com

Link to Game Maker, Audacity, Daz downloads:
http://www.gbalearning.com/gamedevresources.html

Post Modules Files: There is a file that accompanies each Module and shows how the game should look up to, and after, the completion of each module. Also included is an executable file of how the game will look at the completion of each lesson.

Student Starter Files: If a student misses one or more classes, when they return, they can use the Module start file as their "starter" file on the day they return.

Student Tutorials: PDF's of each Student Tutorial Module.

Student Art-Sound Files: All artwork and sound are built-in to the student starter file. This file contains additional artwork if students choose to change the included artwork.

Design Document: This 2-page form sets the student mind working. The students create the design document for a game they would like to create.

STUDENT FILES DISK

Link to Game Builders Academy Web site: http://www.gbalearning.com

Link to Game Maker, Audacity, Daz downloads:
http://www.gbalearning.com/gamedevresources.html

Puggy Sprites: This file contains a number of different sprites that students may load into their games. They were created by Game Builders Academy Master Artist Michael Pugliese. Students may use them Royalty Free.

Backgrounds: For students who wish to concentrate on game design rather than game art, we have included backgrounds that may serve as a Game Intro, Game Win, and Game Lose screens. There are also a variety of other backgrounds the students may wish to import or edit.

Sounds: A variety of sounds which may be loaded into student games.

Buttons: Replay and Start buttons to jump-start the games.

Game Art Appendix: All the artwork shown in Appendix III: Art and Animation for Games. After class discussions, students may want the images to repeat the effects discussed.

Teacher's Manual Table of Contents

continued on following page

Teacher's Manual Table of Contents *Continued from previous page*

continued on following page

TEACHER'S MANUAL TABLE OF CONTENTS *Continued from previous page*

APPENDIXES

Project 1 Overview: Jumper Quest

The game that students will create is referred to as a 2-dimensional, side scrolling, platformer game. There are many games in this genre, the most recognizable being the 2D (2-dimensional) Super Mario Bros. games. "New Super Mario Bros." for the Nintendo DS is one of the more recent examples which students may be familiar with.

Students will create two different levels for their games. The recurring theme while students develop their games will be gravity, and how it affects their game character's world. This is conveyed through their games in a very real and visual way, as programming the player character with gravity is one of the first things students will do. Students will also design and program many other various characters and objects that can either help or hinder the player.

In the first level of the game, the player will have to run and jump from platform to platform in order to reach the goal of the level. Along the way, the player will come across various obstacles that will try to stop him or her from reaching the goal. The first of these obstacles will be the grounder character. As the name suggests, the grounder normally stays on the ground and gets in the way of the player. However, the player can defeat this adversary by jumping on its head. There are also many items the player can use to its advantage, such as springs to reach high places and coins to score points. The player's goal is to reach the portal, which will send him or her to the next room (level) of the game.

The second level of the game can contain all of the previous characters, objects, items, and programming created for the first level, and will also add many new and exciting features. The most recognizable change in this level will be the extended length of the room. Increasing the width of the play area will implement side scrolling functionality, allowing the player to traverse through a much larger environment than the previous level. This level will once again end by reaching the portal, but this time will send you to the win screen. Similar to classic arcade games, a high score table will also be displayed, where players can enter their names and show off their highest scores.

At the end of the course, students will create an "executable" file of their game. This is a true, PC gaming format file, which can then be played on any PC, with or without Game Maker being installed. Students can also post their games on the web or send them via email for friends and family to see and play.

Lesson 1 Overview

Modules 1 - 3

Module 1: Gravity Example

OVERVIEW:

In this Module, students will: Discuss the concept of gravity, and apply it to their games.

Note: Before students start working on the starter file (Module 2), a gravity example file will be opened first as an immediate way for students to see how gravity works in their games. Gravity will be discussed on many occasions throughout the course, and is a core concept of how the player character will behave. The discussion of gravity in this module is intended to bring out the natural excitement students have for science.

STEPS:

1. Navigate to the Student Files folder and open Gravity_Example.gmk.

   Tell Students: The game we will be making is a 2D sidescrolling platformer game, which is similar to the Super Mario games. The player character in our game will be able to run left, right, and jump.

   Ask Students: Speaking of jumping, someone help me out. In real life, when I jump (jump!), can anyone tell me why I fall back down? Why don't I just keep floating up? **Answer**: Gravity.

   Note: Crumple a piece of paper into a ball and hold it in front of you.

   Ask Students: What do you think will happen when I let go of this paper? **Answer**: It will fall.

   Note: Let go of the paper, and let it fall to the ground.

   Ask Students: Why did it fall? **Answer**: Gravity! (let the whole class answer in unison)

   Ask Students: Right! So gravity pulls us down to the ground. If we wanted to be more specific, where does gravity really pull us towards? **Answer**: The center of the Earth.

   Tell Students: Great! Okay, we have an example game set up on your computers. Let's test the game out and see what happens.

2. Test Game! To test the game, either click the Run Game button (green arrow) at top of the Game Maker screen, or select Run>Run Normally, or press the F5 key on the keyboard.

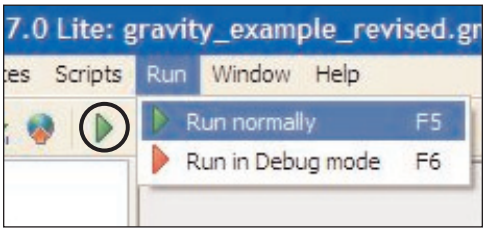
   Ask Students: Something's not right here. What do you think it is? **Answer**: The player isn't falling, and is stuck in mid-air.

   Ask Students: Why do you think the player isn't falling? What do you think is missing in the player world? **Answer**: Gravity.

   Tell Students: Exactly! Now, in our world, we don't need to tell gravity to work. It works automatically. It's not like we could float in the air, and then say: "Alright gravity, start working!", and then fall to the ground!

   Tell Students: However, the computer doesn't know what gravity is, so the game doesn't have gravity automatically like we do. Instead, we have to tell our games to

continued on following page



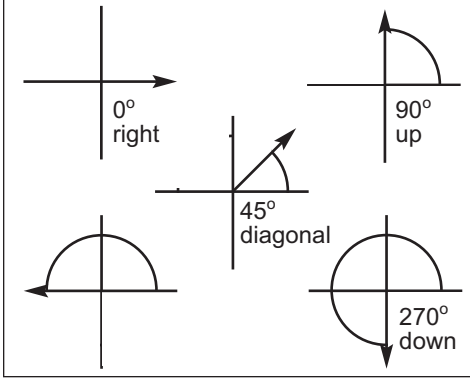
2. To test the game, either click the Run Game button (green arrow) at top of the Game Maker screen, or select Run>Run Normally, or press the F5 key on the keyboard

Fun Gravity Discussion

This module provides a great opportunity for a short, fun discussion on gravity. One (of many) good questions to ask students is this: "In this module, we program the gravity to pull the player down vertically (270), does the force of gravity always pull things down vertically?" The answer is no. The force of gravity pulls objects with lesser mass towards the center of objects with greater mass. So the pull can be in any direction relative to the object being pulled. A good example is a spaceship being drawn towards a planet, moon or star. The spaceship can be pulled in any direction, whichever direction draws it closer to the center of the planet, moon, or star.

Angles and Direction

In Game Maker and other programming environments, direction is based on the angles within a full 360 degree circle. Right is indicated by 0 degrees (360 will work in many programming environments, but in Game Maker, only 0 degrees can be used to specify right as a direction). Left is indicated by 180 degrees; Up is indicated by 90 degrees; Down is indicated by 270 degrees. All incremental angles between 0 and 359 are usable - this allows very fine control of direction of motion in game programming.



Module 1: Continued from previous page

have gravity. Always remember that unless we tell the computer to do something, it will not do it. So, let's all become game programmers and program gravity into our games!

3. In the Library, double-click the player object to bring up the Object Properties window for the player.

   Note: Make sure the object being programmed is the player object. The game has two objects, so it's important to make sure students are always aware of which object they are working on. (See sidebar on page 12, "What Object are We Currently Programming?")

4. In the Object Properties window, left-click the Add Event button (this will bring up the Event Selector).

   Brief Discussion: This is the introduction of event-driven programming. Explain to students that event-driven programming means programming is based on an event occurring, and that event causes one or more actions to occur. For example, an event can be a key being pressed, and the resulting action can be an object moving in a specific direction.

5. In the Event Selector window, left-click Create.

6. On the right side of the Object Properties window, in the Actions panels, Left-click the Move tab (top right corner), and then right-click the Set Gravity button (this will bring up the Set Gravity window).

7. Set the options in the Set Gravity window:

   Ask Students: In which direction does gravity pull us? **Answer**: Down

   Tell Students: For now, I'll tell you that to say "down", we have to put 270 next to direction. I will explain later why this is, but for now, let's just put 270 for direction.

   a. For Applies to, put Self

   b. For direction, put 270

   c. For gravity, put 1

   d. the Relative box remains unchecked

   e. Click OK

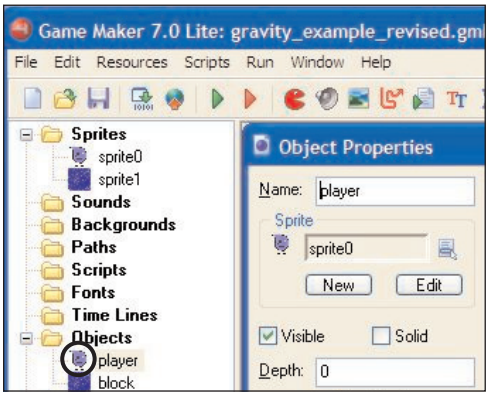
8. Test game! Player should fall, but not stop at the blocks. In other words, player will fall through the blocks.

   Remind Students: We're about to try something new - what do we expect to happen? Expect it not to work. If it works, that's gravy.

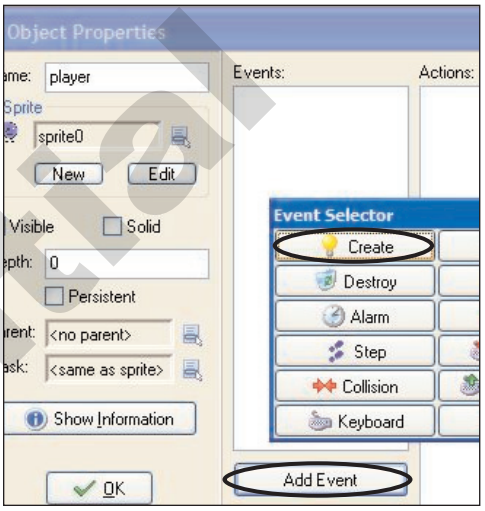
9. In the Object Properties window, left-click the Add Event button (this will bring up the Event Selector window).

   Ask Students: Which event do you think we should choose? **Answer**: Collision.

continued on following page

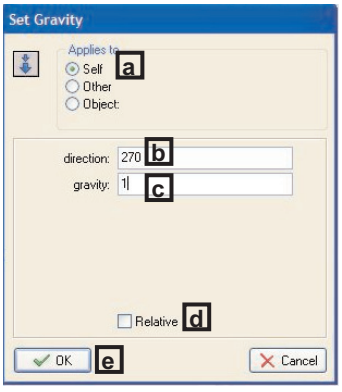


3. In the Library, double-click the player object



4,5. Left-click the Add Event button, then in the Event Selector, left-click Create

6. Left-click the move tab, then right-click the Set Gravity button



7. Set Applies to Self, direction to 270, gravity to 1, leave Relative unchecked.

Module 1: Continued from previous page

10. In the Event Selector window, left-click Collision and choose block from the flyout menu (since we are currently programming the player object, the other object in the collision is the block, so we choose block from the Collision flyout menu).

11. On the right side of the Object Properties window, in the Actions panels, left-click the Move tab (top right corner), and then right-click the Set Gravity button (this will bring up the Set Gravity window).

12. Set the options in the Set Gravity window:

    a. For Applies to, put Self

    b. For direction, put 270

    c. For gravity, put 0

    d. the Relative box remains unchecked

    e. Click OK

13. Test game!

Remind Students: We're about to try something new - what do we expect to happen? Expect it not to work. If it works, that's gravy.

END 

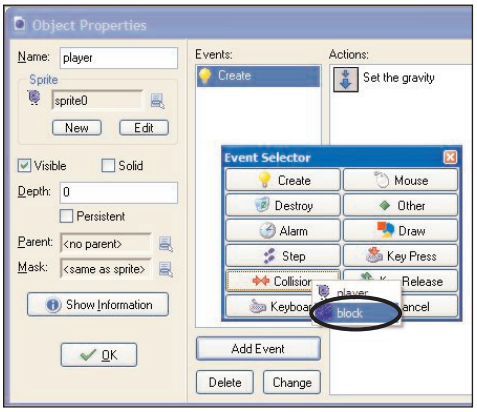
VOCABULARY:

Gravity: The force of attraction by which all things in the atmosphere are pulled toward the center of the earth.

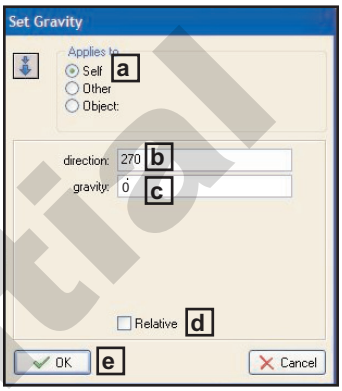
Program: A series of instructions; a sequence of events.

Collision: (in context of programming): A collision event is when any two objects touch each other in any way, at any speed.

Debug: To detect and remove programming errors.



10. Left-click Collision and choose block from the flyout menu



12. Set direction to 270, gravity to 0, and leave Relative unchecked

Module 2: Open and Save Student_Starter File

OVERVIEW:

In this Module, students will: Open the Student starter file, and save their games.

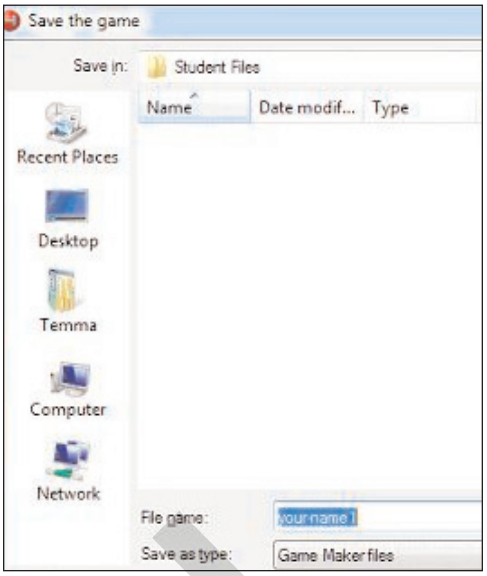
Introductory Discussion:
Students are given a Student starter file containing many necessary resources the students will need to create their 2-dimensional side-scrolling platform game.

Note: Students must open the game file, "student_starter.gmk", which contains many necessary resources and programming needed for students to continue developing their games. Some of this programming is used to control certain things in the game "behind the scenes," so neither teacher nor student need to pay any attention to it, as it has no effect on the lessons throughout the course. Students will save and open their own game files for the rest of the course, and will not have to use any more starter files after this.

STEPS:

1. Navigate to the Student Files folder and open student_starter.gmk.
2. Using the File menu, left-click on Save As.
3. In the Save the Game window, navigate to your designated folder and name your file your-name1, replacing "your-name" with your first and last name.

END



3. Save the game into your designated folder, and use your name to name the file

Module 3: Create the Sprite (Image) for the Player Object

OVERVIEW:

In this Module, students will: Create and draw a sprite for their avatar (game player character or main character) using the tools in the built-in Image Editor.

Note: The Image Editor is a built-in drawing tool included in the software. Students will be using the Image Editor to draw their own characters, items, and obstacles throughout the course. By adding their own artwork, each student's game will have a unique and personalized look. Encourage students to start out with simple drawings -- especially for any who are having trouble.

STEPS:

1. In Game Maker, from the top menu choices, choose Resources>Create Sprite (this will bring up the Sprite Properties window).

   Brief Discussion - sprites and objects: A sprite is an image that is used for an item or character in the game. A sprite doesn't actually get programmed. An object gets programmed (you will create an object in the next Module). A sprite is an image that goes on the object so that the object, which can be programmed, looks a certain way. A good analogy is: A sprite is like a suit of clothes, an object is like the person that wears the suit of clothes. If you want to tell a person to do something, you give instructions to the person, not the clothes. In programming, you give instructions to an object, not a sprite. But you need to put a sprite on the object so it can look like something (a robot, a boulder, a car, etc.). Without a sprite assigned to the object, the object will not be visible in the room when the game is being played.

2. In the Sprite Properties window, click the Edit Sprite button (this will bring up the Sprite Editor window).

   Note: Since we're now creating our own sprite from scratch, we click Edit Sprit so we can access the Sprite Editor where we can "paint" our own sprites.

3. In the Sprite Editor window, click the Pencil icon, or double-click directly on the sprite image icon (this will bring up the Image Editor window).

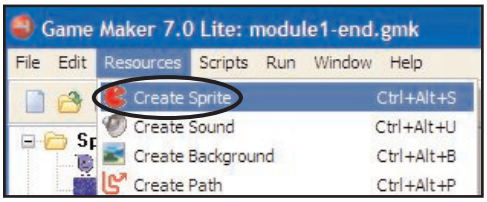
4. In the Image Editor window, click on the magnifying glass (a few times) to zoom in on the art as you work (this does not affect the size of the image - it only affects the preview as you work).

   Note: The following steps will show you how to draw a simple character using the pencil tool in the Image Editor. Although these steps are specific, encourage students to experiment if they wish to draw something different. Anything the students create is fine, as it is purely visual and will not affect the mechanics of the game. If you would like to learn more about the different tools and about drawing in game maker, such as filling in the shapes with color, refer to Appendix III: Art and Animation for Games. It is recommended that you familiarize yourself with the Image Editor as much as possible, which will make it that much easier to help students.

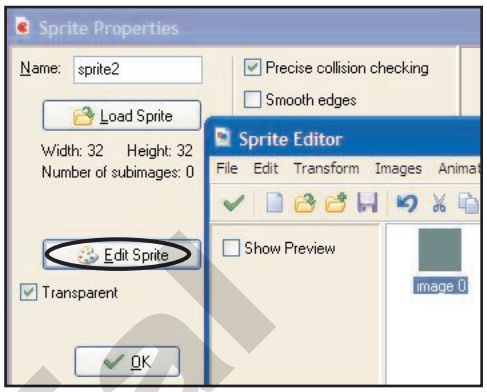
5. In the Image Editor window, in the tools section on the left, click the pencil tool (even though it may already be selected).

6. In the colors section on the right, select any color by left-clicking it. This will be the color you draw with.

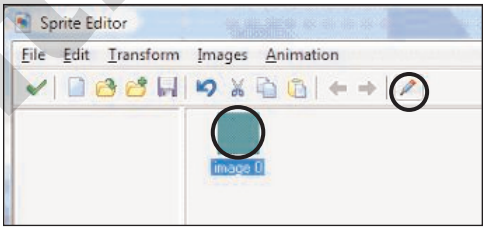
continued on following page



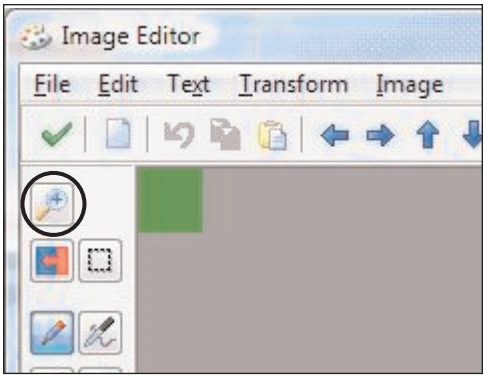
1. Choose Resources>Create Sprite



2. Click the Edit Sprite button



3. Double-click directly on the sprite image icon (or click the pencil tool)



4. Click on the magnifying glass (a few times) to zoom in on the artwork

Module 3: Continued from previous page

Note: Since the background of the game is gray by default, choosing gray as a drawing color may make the player appear invisible against the background. It would be best to choose any color other than gray.

7. In the drawing area, click and drag to start drawing a circle or egg shape as shown at right.

   Note: Try to leave some room on the bottom of the image for the feet

   Tell Students: "Don't worry about making it perfect. Just try your best to get a circle shape. You can always change your sprites later on too."

   Note: Be sure to check how students are doing at this point, and give individual help where needed, as this may be the first time they are drawing on a computer.

8. Draw in facial features, such as eyes and a mouth, in the circle as shown at right.

9. Draw two feet under the circle shape. Try drawing them as two letter "L"s as shown at right.

   Note: If time allows, you may want to give students some time to add more features to the drawing.

10. In the top left corner of the Image Editor window, click the green check-mark. (This will close the Image Editor window.)

11. At the top left of the Sprite Editor window, click the green check mark (this will close the Sprite Editor window).

12. In the Library, double-click the player object (this will bring up the Object Properties window for the player object).

13. In the Object Properties window for the player object, left-click the "famous add-thingy button" and choose the new sprite from the flyout menu, then click OK.

14. In the Object Properties window, click OK (this will close the Object Properties window).

15. In the Library, double-click level1 to bring up the Room Properties window. The student's player sprite is now in the room.

END

Vocabulary:

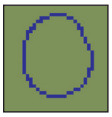
Sprite: An image used for an object or character in a game.

Object: Something that can be programmed (given instructions).

Avatar: Specifically, the sprite used as the player's character in a game.

Image Editor: Built-in drawing program included in the Game Maker software; used to create artwork for characters, backgrounds, and any other type of graphics for the game.

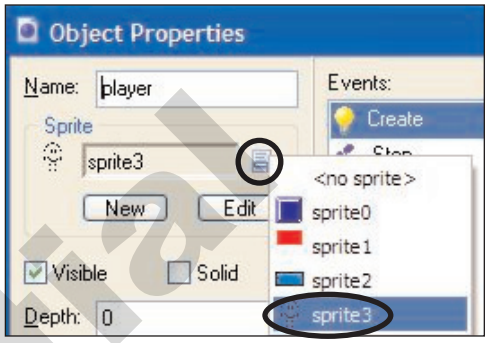
Graphic: A visual representation; an image.



7. In the drawing area, click and drag to start drawing a circle or egg shape

8. Draw in facial features, such as eyes and a mouth, in the circle

9. Draw two feet under the circle shape. Try drawing them as two letter "L"s as shown



13. Left-click the "famous add-thingy button" and choose the new sprite from the flyout menu

The "Famous Add-Thingy Button"

The button referred to in Step 13 (at left, that you need to click in order to assign a sprite to an object, has no official name in Game Maker. When we started teaching these classes, we needed a way to refer to this button so the students would remember it. We named it the "famous add-thingy button" (because, in a number of places within Game Maker, this button is used to "add" something - in this case, it is used to add a sprite to an object). Additionally, when the term "famous add-thingy button" is said with some drama or humor, students remember this button very well, and younger students have a good deal of fun just saying it.

NOTES

Lesson 2 Overview

Modules 4 - 8

Module 4: Program Player Character Movement

OVERVIEW:

In this Module, students will: Be introduced to event-driven programming; Program their game character to move right and left via keyboard input from the player; Be introduced to basic graphing principles to control programmed movement.

Introductory Discussion:

Note: This is the introduction of event-driven programming. Event-driven programming is based on an event occurring, and that event causes one or more actions to occur. For example, an event can be a key being pressed, and the resulting action can be an object moving in a specific direction. This is how all programming throughout the course will be done. This module also presents the first real application of math in the course. While vertical movement is left to gravity in conjunction with jumping (which will be programmed in an upcoming module), horizontal movement will be programmed using the Cartesian coordinate graphing system. Specifically, students will use the x-axis to move the player left and right. The discussions and steps will illustrate how this is applied to the game. First though, as a refresher, have a brief review of how gravity works, and ask students what they think is missing from their games.

Review with Students: Take a moment to review how gravity works in real life and in the game using the following discussion questions.

Ask Students: When we test our game, what is pulling the player down? **Answer**: Gravity.

Ask Students: How is this happening? Did it work automatically? **Answer**: No. We programmed it to happen.

Ask Students: What else does the player need to be able to do? **Answer**: Move left and right, jump.

STEPS:

1. In the Library, double-click the player object to bring up the Object Properties window for the player.

   Note: Make sure the object being programmed is the player object. It's important to make sure students are always aware of which object they are working on. Check the object name field to be sure. (See sidebar on the right, "What Object are We Currently Programming?")

   Discuss with Students: If and Then Statements -- There are two important words that we will need to remember in order to program our games. The first word is "if" and the second word is "then." Let's use "if" and "then" to make up a sentence. For example, "if it is raining...then I will get an umbrella"….If I am hungry, then I will eat (any other real world examples you come up with can be used as well).

   Ask Students: Can anyone give me another "if and then" sentence? It can be about anything!

   Note: Just about anything students come up with will be correct. If they are having trouble, give them one to start off with such as "if I am hungry" or "if I am tired" and then let them finish the sentence for you. Take this discussion as far as you feel is appropriate.

continued on following page

What Object are we Currently Programming?

Students will often think they are programming (or working on) one object, when they are actually programming another. This can, of course, lead to significant problems.

To avoid this confusion, it's important to consistently remind students to make sure they are programming the correct object. The single best way to know which object is currently being programmed is to check the name field of the Object Properties window.

To reinforce this in students, frequently ask students (at random) "What object are we currently programming?" The answer is whichever object's name appears in the object name field. For this particular module, students should be working on (programming) the controller object.

Keyboard and Keypress

For left and right movement, we use the Keyboard events. For jumping, however, we use the Keypress event. The difference is this: The Keyboard event is a repeating event (the event occurs repeatedly as long as the key is held down on the keyboard). The Keypress event is a one-time event (the event occurs only once when the key is pressed on the keyboard - to repeat the event, the key must be released and pressed again), this is what we want for our jumping functionality.

Module 4: Continued from previous page

Ask Students: Can anyone give me an "if and then" sentence for moving the player character? **Answer**: If I press the right arrow key, then the player will move to the right.

2. In the Object Properties window, left-click the Add Event button (this will bring up the Event Selector).

   Ask Students: Which one of these events might help us? **Answer**: Keyboard (There are many other choices that sound like they may work, such as Step, Key Press, Key Release, Mouse etc., so encourage students to voice all answers, but let them know that Keyboard is the one we will use right now).

3. In the Event Selector, left-click the Keyboard button (this will bring up the Keyboard events menu).

   Note: In the Events Selector, there are two similar, but very different, events: Keyboard and Key Press. The difference is: Keyboard is a repeating event (press the given key and the programmed action occurs repeatedly - as long as the key is held down), Key Press is a one-time event (press the given key and the programmed action occurs only once).

4. In the Keyboard events menu, choose <Right> (this is the event that occurs when the Right Arrow key is pressed on the keyboard). The <Right> event should now appear in the player object Events list.

5. On the right side of the Object Properties window are the Action panels (tabs for sections and buttons for actions within each section). Left-click the Move tab (top right corner), and then right-click the Jump to Position button (this will bring up the Jump to Position window).

   Note: The term "jump," as it is used in Game Maker, actually means "move." Even though there are other actions that are specifically called "move," we will use the "Jump to Position" action here. Jump to Position means move from one position (graph point) to another.

   Discuss with Students: X- and Y- axes

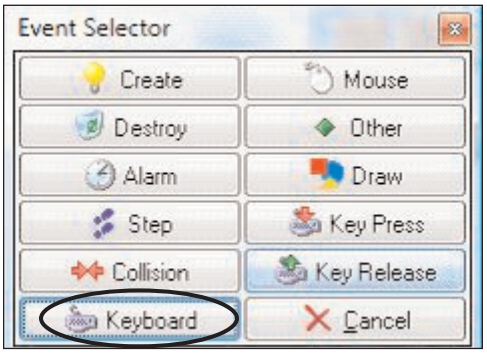
   Important Note: Students may or may not have heard about graphing in school yet. Assume that you are preparing them for when they see graphing in school. Here is one suggestion to implement the introductory graphing discussion:

   Discuss with Students: Without mentioning anything about graphs or math at all, draw a big plus sign(x and y axes) on the board with no other markings and ask students what they think it looks like. Almost any answer they come up with will be correct in some way (ex: plus sign, corners, angles, crosshair) so encourage and be excited about any answers students voice.

   At this point, assure students that all their answers were correct. Then ask students if they have ever heard of a graph. If they have, you can ask them about the following. If not, lead them through the following:

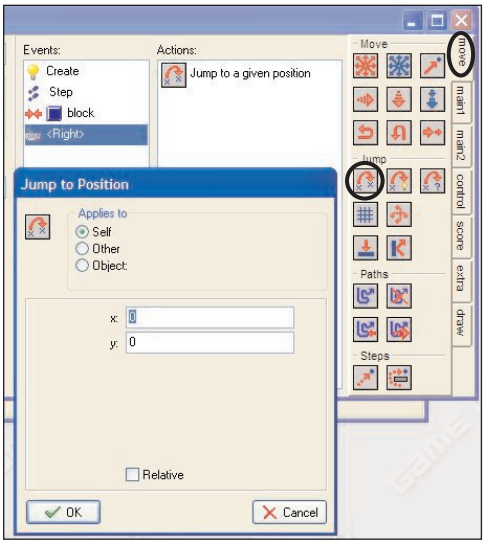
   Explain that the line going left and

continued on following page



3. Left-click the Keyboard icon

4. Left-click the Keyboard icon, then choose <Right> from the flyout menu



5. Left-click the move tab, then right-click the Jump to Position button

Module 4: Continued from previous page

right is the x-axis. Draw an x on the graph. (refer to diagram Figure1 for proper placement)

Explain that the line going up and down is the y-axis. Draw a y on the graph (refer to Figure1 for proper placement).

Review with Students: Reinforce this with hand motions for left and right (x) and up and down (y). Have students mimic horizontal and vertical hand movements as well.

Ask Students: Keeping in mind what we've just learned, which axis should we use to move the player right? **Answer**: x-axis.

Tell Students: Explain to students that the right side of the x-axis has positive numbers (emphasize positive). Draw 1, 2, 3, 4, etc in the proper place on the x-axis. (refer to Figure1 for proper placement).

Ask Students: If the right side is positive, can anyone guess what the numbers on the left are? **Answer**: Negative.

Ask Students: What symbol usually comes before a negative number? **Answer**: The minus symbol. Draw -1, -2, -3, -4, etc in the proper place on the x-axis (refer to Figure1 for proper placement)

Ask Students: Keeping in mind what we've just learned, if we want to move the player to the right, do you think we would use a positive number, or a negative number for x? **Answer**: Positive.

6. Set the options in the Jump to Position window:

   a. Applies to Self remains the default choice

   b. Set the x movement to positive 5 (can be written as 5 or +5).

   c. At the bottom of the Jump to Position window, Click the Relative checkbox

   d. Click OK

   Note: When Relative is checked, the next new x (and/or y) position an object moves to is based on the position relative to the object's current position (i.e. +4 means move 4 units to the right from the current position. If an object's current position is x=6, and it moves x+4 relative, it's new position is x=10). When Relative is not checked, the next new x (and/or y) position an object moves to is based on the position in absolute graph coordinates (i.e. +4 means move to x=4 no matter what the current position of the object. If an object's current position is x=6, and it moves x+4 absolute [not relative], it's new position is x=4, not x=10). This will be a common stumbling point for many students. It's an easy fix, but highly frustrating for students until they understand it. For now, don't worry about explaining this to the students as it may be overwhelming at this point. Simply instruct students to make sure Relative is checked for this Module.

7. Test Game! To test the game, either click the Run Game button (green arrow) at top of the Game Maker screen, or select Run>Run Normally, or press the F5 key on the keyboard.

continued on following page

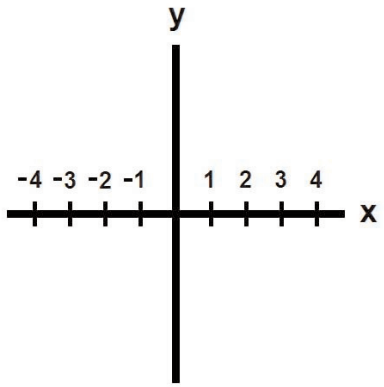
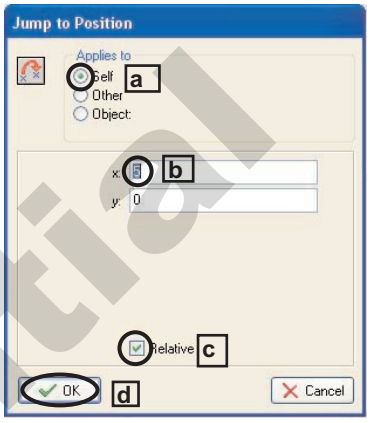


Figure1. Use this sample to help explain graphing and x and y axes



6. Leave Applies to on Self, Set the x value to 5, leave the y value at 0, and check the Relative checkbox

Which Way is Up?

In math, y values increase as you go further **up** the y-axis. But in programming, y values increase as you go further **down** the y-axis. Additionally, in programming, the origin of a graph (computer window area) is in the upper left corner, as opposed to in the center as it is in traditional mathematics.

The main reason programmers set y values to function this way (long ago), and set the origin in the upper left is because with that setup, the entire visible area (game play area, program window, etc.) is in positive x and y values. It's simply easier to work with the programming area in positive values. An absolute negative value (x or y) indicates that the position is above or to the left of the screen area.

Note: x-values work the same in math and in programming, that is, increase when going further right, and decrease when going further left.

Module 4: Continued from previous page

Remind Students: We're about to try something new - what do we expect to happen? Expect it not to work. If it works, that's gravy.

8. Debug as Necessary (click the x in the upper right corner of the game window to close the game play mode and return to the game design mode to debug and/or continue working). This is included as a Step in itself, because it is expected that much of the time, debugging will be necessary when adding new programming instructions or game features (see sidebar on the right, "Very Important Note About Testing Games").

   Important Discussion on Programming Movement: At this point in the project, it is an excellent time to discuss x and y planes and coordinates, and other graphing principles. Take this discussion in whatever directions are best for your students at their given level.

9. In the Object Properties window, left-click the Add Event button (this will bring up the Event Selector).

10. In the Event Selector, left-click the Keyboard button (this will bring up the Keyboard events menu).

11. In the Keyboard events menu, choose <Left> (this is the event that occurs when the Left Arrow key is pressed on the keyboard). The <Left> event should now appear in the player object Events list.

12. On the right side of the Object Properties window are the Action panels (tabs for sections and buttons for actions within each section). Left-click the Move tab (top right corner), and then right-click the Jump to Position button (this will bring up the Jump to Position window).

13. Set the options in the Jump to Position window:

    a. Applies to Self remains the default choice

    b. Set the x movement to negative 5 (must be written as -5).

    c. At the bottom of the Jump to Position window, Click the Relative checkbox

    d. Click OK

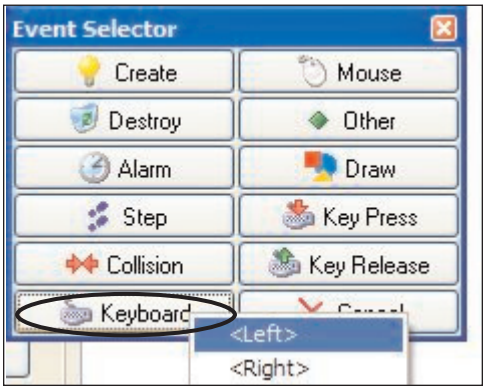
14. Test game!

    Remind Students: We're about to try something new - what do we expect to happen? Expect it not to work. If it works, that's gravy.

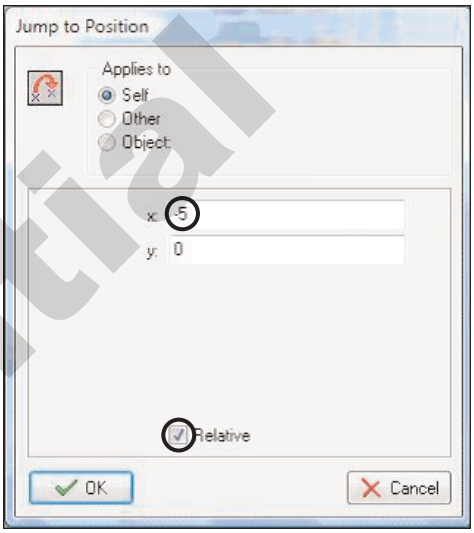
END

Vocabulary:

See end of next module (module 5) for module 4 vocabulary.



10,11. Left-click the Keyboard icon and choose <Left> from the flyout menu



13. Set the x value to -5, leave the y value at 0, and check the Relative checkbox

Very Important Note About Testing Games

Before you have students test their games, remind them: "**What do we expect to happen when we try something new for the first time? Expect it not to work.** If it works that's gravy (bonus). Whenever we try something new expect it not to work the first time. But be sure that you will get it to work, eventually. Maybe not the first time, maybe not the second time, maybe not the hundredth time - but it will work if students stick with it.

Module 5: Program Player Character Jumping

OVERVIEW:

In this Module, students will: Program their game character to jump in the air via keyboard input from the player.

Introductory Discussion: Giving the player character the ability to jump allows the player to reach new areas in the level. Jumping, in conjunction with moving left and right, will be the primary source of motion for the player. In addition, because gravity was previously programmed, the player will jump and then fall accordingly, without additional programming.

STEPS:

1. In the Library, double-click the player object to bring up the Object Properties window for the player.

   Note: Make sure the object being programmed is the player object. It's important to make sure students are always aware of which object they are working on. Check the object name field to be sure. (See sidebar on page 12, "What Object are We Currently Programming?")

   Ask Students: Give me an "if then" sentence for making the player jump. **Answer**: If I press the up arrow key(or any other button), then the player will jump

2. In the Object Properties window, left-click the Add Event button (this will bring up the Event Selector).

3. In the Event Selector, left-click the Key Press button (this will bring up the Key Press events menu).

   Note: Be sure that students use Key Press, and not Keyboard. (See sidebar on page 12, "Keyboard and Key Press?")

4. In the Key Press events menu, choose <Up> (this is the event that occurs when the Up Arrow key is pressed on the keyboard). The <Up> event should now appear in the player object Events list.

   Note: Lead students to the answers in the discussion below if they are having trouble.

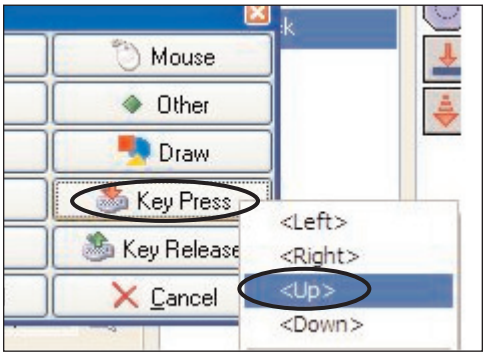
   Ask Students: Can anyone give me a big word that starts with a "V" that means up and down? **Answer**: Vertical (if a student answers "y", encourage and praise the student for having the right idea, and remembering that the y-axis controls up and down)

   Ask Students: Can anyone give me the big word for left and right? This one starts with an "H". **Answer**: Horizontal

   Ask Students: So, we know that vertical is for up and down, and horizontal is for left and right. Which one do you think would be best to describe jumping? **Answer**: Vertical

5. On the right side of the Object Properties window, in the Actions panels, left-click the Move tab (top right corner), and then right-click the Speed Vertical button (this will bring up the Speed Vertical window).

continued on following page



3,4. Left-click the Key Press icon, then choose <Up> from the flyout menu

Module 5: Continued from previous page

6. Set the options in the Speed Vertical window :

   a. For Applies to, put Self (default option)

   b. For vert.speed, put -15

   c. the Relative box remains unchecked

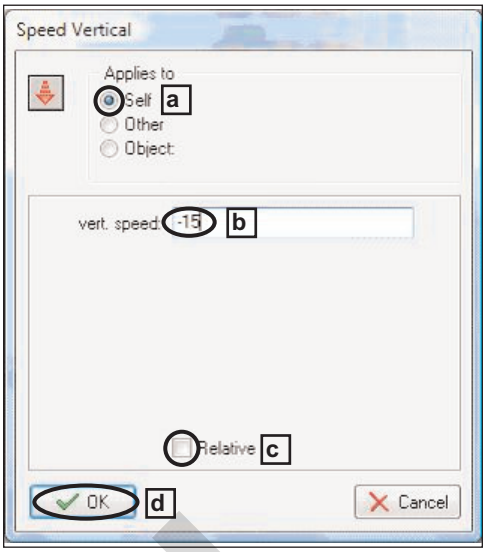
   d. Click OK

7. Test game!

   Remind Students: We're about to try something new - what do we expect to happen? Expect it not to work. If it works, that's gravy.

8. Debug as necessary.

END



6. Leave Applies to on Self, set the vert. speed to -15, leave the Relative box unchecked

Module 5 Vocabulary

Gravity: The force of attraction by which all things in the atmosphere are pulled toward the center of the earth.

Vertical: Pertaining to up and down directions (y-axis).

Horizontal: Pertaining to Left and right directions (x-axis).

Module 4 vocabulary

Gravity: The force of attraction by which all things in the atmosphere are pulled toward the center of the earth.

Program: A series of instructions; a sequence of events.

Event-driven programming: Programming that is based on an event occurring, and that event causing one or more actions to occur.

Object-Oriented Programming: Computer programming based on giving instructions to objects.

Cartesian Coordinate (Graphing) System: System invented by Rene Descartes to indicate position in two-dimensional space, based on x and y (horizontal and vertical) positions on a grid or graph.

x-axis**,** y-axis**:** Horizontal and vertical planes (respectively) of the Cartesian Coordinate System.

Event: An occurrence that triggers an action.

Action: An occurrence that takes place in response to an event.

Positive number: Any number greater than 0.

Negative number: Any number less than 0; marked with a minus symbol (-) before the number.

Horizontal: Pertaining to left and right directions (x-axis).

Vertical: Pertaining to up and down directions (y-axis).

Module 13: Create the Sprite (Image) for the Spring Object

OVERVIEW:

In this Module, students will: Create and draw a sprite for a new item, called the spring, using the tools in the built-in Image Editor.

Introductory Discussion : We will be adding on an additional feature for the game -- the spring -- which will launch the player up in the air with a much stronger force than regular jumping, allowing the player to reach greater heights in the level. The drawing lesson of the spring goes into some more in-depth (and fun) animation techniques.

STEPS:

Tell Students: Today we will be working on a brand new feature for your game that will help the player. We are going to make a spring.

Ask Students: Has anyone ever seen a spring in games you played at home? What does a spring do? **Answer**: A spring bounces the player up high in the air.

Ask Students: We will draw and animate the spring. Where do I go if I want to draw something? **Answer**: Create Sprite

Ask Students: We need a new sprite - what is the first thing we do? **Answer**: Choose Resources>Create Sprite

1. Choose Resources>Create Sprite (this will bring up the Sprite Properties window).
2. In the Sprite Properties window, click the Edit Sprite button (this will bring up the Sprite Editor window).

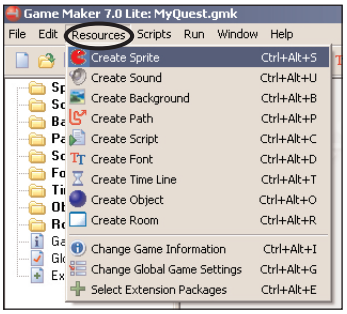
   Tell Students: Since we're now creating our own sprite from scratch, we click Edit Sprit so we can access the Sprite Editor where we can "paint" our own sprites.
3. In the Sprite Editor window, click the Pencil icon, or double-click directly on the sprite image icon (this will bring up the Image Editor window).
4. In the Image Editor window, click on the magnifying glass (a few times) to zoom in on the art as you work (this does not affect the size of the image - it only affects the preview as you work).

   Note: For this drawing lesson, the teacher will draw first, and then have the students follow the step-by-step directions that follow.

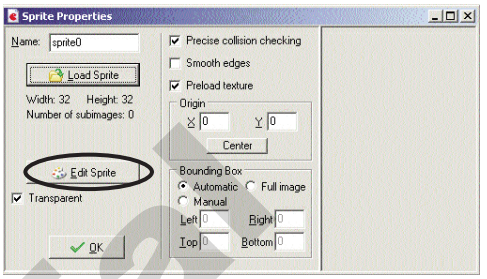
   Tell Students: First we will draw the top of the spring. Watch me first, and then follow along.
5. Draw something like the image here… 
6. In the upper left corner of the Image Editor window, click the green check mark (this will close the Image Editor window and return you to the Sprite Editor window).

   Tell Students: We will be animating this spring. Instead of drawing this image over, let's copy the image to save us some work.

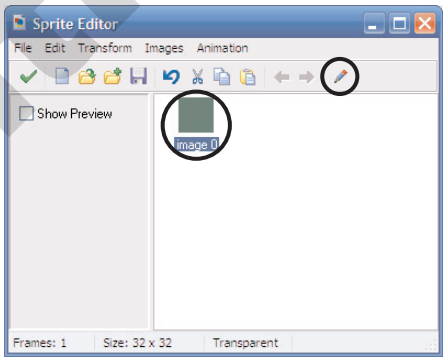
continued on following page



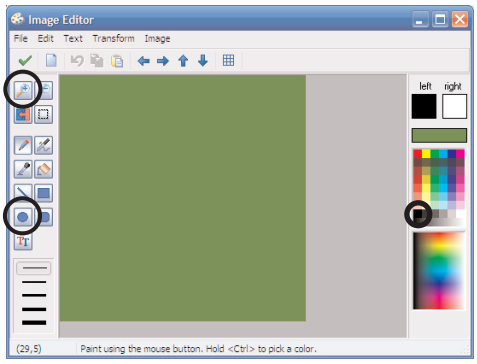
1. Choose Resources>Create Sprite



2. Click Edit Sprite



3. Click the Pencil icon or double-click directly on the sprite image icon



4. Click the magnifying glass to zoom in on the art as you work, left-click to select the Draw an ellipse tool, left-click to select a color

Module 13: Continued from previous page

7. In the Sprite Editor window, choose Animation>Set Length (this will bring up the Animation Length window).

8. In the Animation Length window, set the length to 3 frames, click OK.

   Note: You should now see three images in the Sprite Editor window - these are your original image and two duplicates - these are the frames of the animation.

   Ask Students: How many frames, or images, are there now? **Answer**: 3

9. In the Sprite Editor window, double-click image1 (the second image) to bring up the Image Editor window for that particular image.

   Tell Students: We want to make the spring look like it is springing up. Let's move this spring image up a little.

10. In the Image Editor window, click the blue up arrow 5 times to move the image up 5 times.

11. In the upper left corner of the Image Editor window, click the green check mark (this will close the Image Editor window and return you to the Background Properties window).

    Tell Students: Now let's work on the last frame and make it even higher!

12. In the Sprite Editor window, double-click image2 (the third image) to bring up the Image Editor window for that particular image.

13. In the Image Editor window, click the blue up arrow 10 times to move the image up 10 times.

14. In the upper left corner of the Image Editor window, click the green check mark (this will close the Image Editor window and return you to the Sprite Editor window).

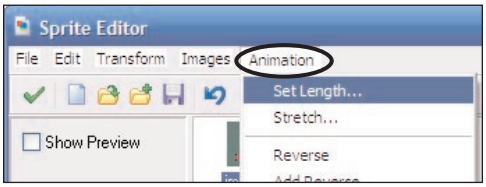
    Tell Students: Great! Now that we've finished the main top part of the spring, let's go back in and add the bottom springy part to it.

15. Open up each frame starting with the first (image0) and draw the bottom coil part of the spring using the pencil tool as shown in the images below (see step 12 above for how to open a single image):

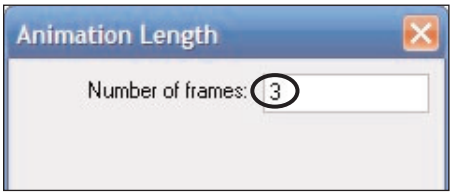
image 0 image 1 image 2

Tell Students: Let's see what the animation looks like!

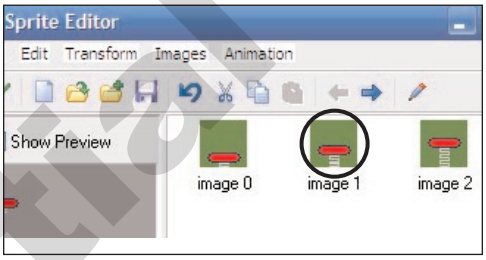
continued on following page



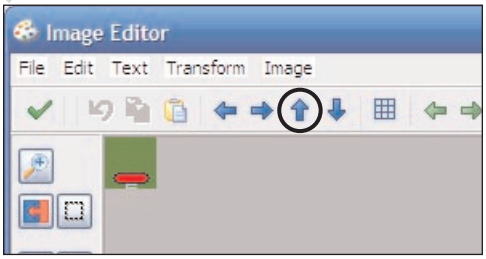
7. Choose Animation>Set Length



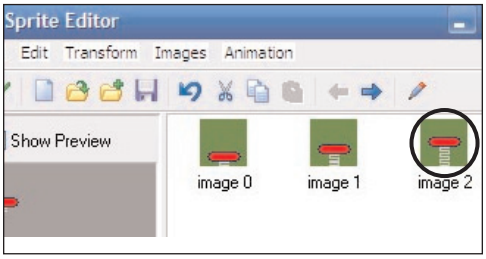
8. Set the Number of frames to 3



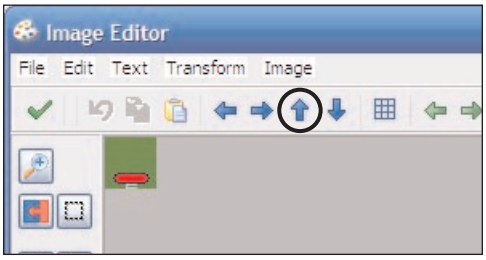
9. Double-click image1



10. Click the up arrow five times



12. Double-click image2



13. Click the up arrow ten times

Module 13: Continued from previous page

16. When all three frames are done, in the Sprite Editor window, click the Show Preview checkbox to preview the animation (preview speed can be changed in the Speed option - though this is for preview purposes only. It does not necessarily represent the speed at which the animation will run in the game - that is programmed separately).

    Tell Students: Great work! Now I'll show you a little trick to make the animation look even smoother!

17. In the Sprite Editor window, choose Animation>Add Reverse (this will duplicate the existing frames and paste them into the Image Editor in reverse).

18. In the upper left corner of the Sprite Editor window, click the green check mark (this will close the Sprite Editor window and return you to the Sprite Properties window).

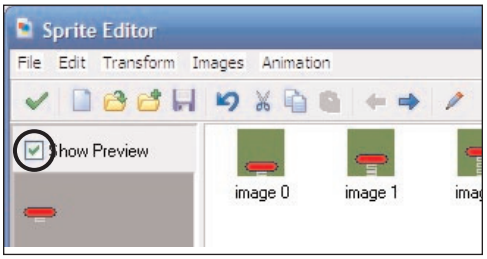
END

VOCABULARY:

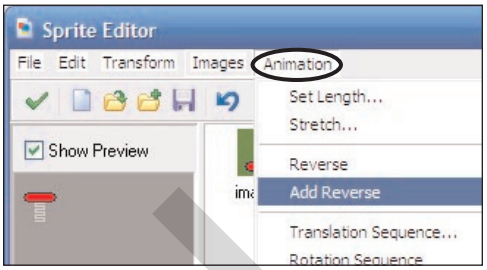
Sprite: An image used for an object or character in a game.

Frame (Image): A single image in an animation sequence; when multiple frames are quickly displayed one after another, this creates animation.

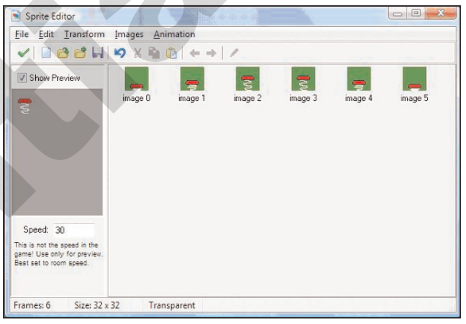
Cell-by-cell (frame-by-frame) animation: The process of animating a series of drawings by running them quickly one after another, in the same physical position, so they appear to be as one moving image. This is the principle behind movie film and cartoon animation, among other uses for this technique.



16. Click the Show Preview checkbox



17. Choose Animation>Add Reverse



After step 17, you should see the new frames in the Sprite Editor window

Module 14: Create the Spring Object and Program Player's Collision with Spring

OVERVIEW:

In this Module, students will: Create an object for the spring; Program the player to launch up on collision with the spring; Place spring object in the room.

Ask Students: Now that we've finished the sprite, what do we need? **Answer**: Object

STEPS:

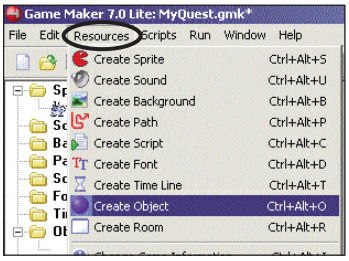
1. In Game Maker, from the top menu choices, choose Resources>Create Object (this will bring up the Object Properties window).
2. In the Object Properties window, next to Sprite (<no sprite>) left-click the "famous add-thingy button" and select the spring sprite from the flyout menu. This will assign a sprite to the object.
3. In the upper left of the Object Properties window, in the Name field, type a name for the object. Name this particular object "spring."
4. In the Object Properties window, click OK (this will close the Object Properties window). You should now see your new object in the Library, in the Objects folder.
5. In the Library, double-click the player object to bring up the Object Properties window for the player.
6. In the Object Properties window, left-click the Add Event button (this will bring up the Event Selector).

   Ask Students: What is it called when two things touch or bump into each other? **Answer**: Collision
7. In the Event Selector window, left-click Collision and choose spring from the flyout menu (since we are currently programming the player object, the other object in the collision is the spring, so we choose spring from the Collision flyout menu).

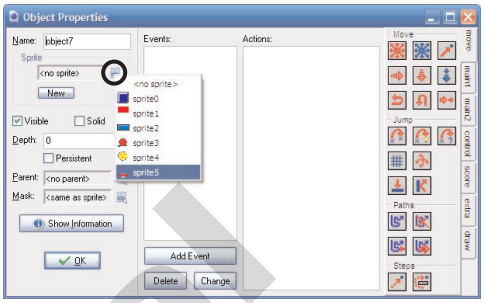
   Ask Students: So, if the player collides with a spring, the player will launch up in the air. Have we ever done programming like this before -- where the player is launched in the air? **Answer**: Yes, when we programmed the player to jump.

   Ask Students: Someone tell me, did we move the player vertically or horizontally? **Answer**: Vertically
8. On the right side of the Object Properties window, in the Actions panels, left-click the Move tab (top right corner), and then right-click the Speed Vertical button (this will bring up the Speed Vertical window).

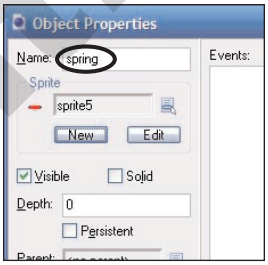
continued on following page



1. Choose Resources > Create Object



2. Left-click the "famous add-thingy button" and choose the spring sprite from the flyout menu



3. Name the object "spring"

7. Left-click the Collision icon and choose spring from the flyout menu

8. Left-click on the move tab, then right-click on the Speed Vertical icon

Module 14: Continued from previous page

9. Set the options in the Speed Vertical window:
    a. For Applies to, select Self (the default option)
    b. For vert.speed, put -20
    c. the Relative box remains unchecked
    d. Click OK
10. In the Library, double-click level1 to bring up the Room Properties window.
11. In the upper left of the Room Properties window, click the "Objects" tab.
12. In the lower left of the Room Properties window, next to the field labeled "Object to add with left mouse," left-click the "famous add-thingy button." This will bring up a flyout menu of all objects currently available. Select the spring object from the flyout menu.
13. Position the cursor in the large open area of the Room Properties window and left-click to add the selected object (spring). Add 5 spring objects to the room above the platform blocks.
14. Test game!

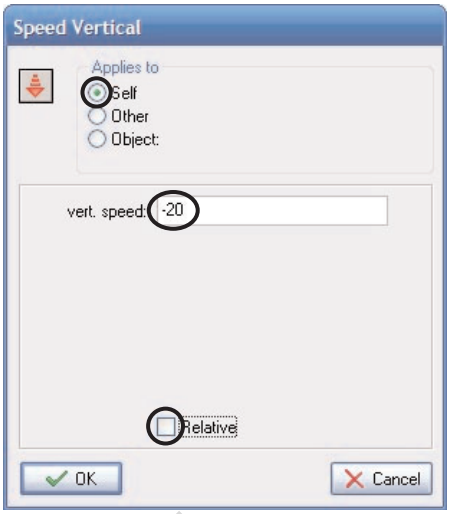
    Remind Students: We're about to try something new - what do we expect to happen? Expect it not to work. If it works, that's gravy.

END 

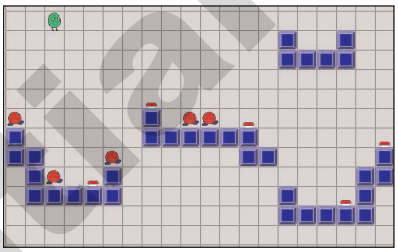
Vocabulary:

Collision (in context of programming): A collision event is when any two objects touch each other in any way, at any speed.

Vertical: Pertaining to up and down directions (y-axis).



9. Leave Applies to set to Self, set vert. speed to -20, leave Relative unchecked

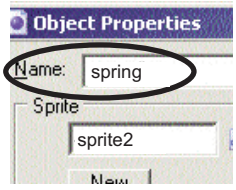


13. Add five spring objects to the room, above the platform blocks

What Object are we Currently Programming?

Students will often think they are programming (or working on) one object, when they are actually programming another. This can, of course, lead to significant problems.

To avoid this confusion, it's important to consistently remind students to make sure they are programming the correct object. The single best way to know which object is currently being programmed is to check the name field of the Object Properties window.



To reinforce this in students, frequently ask students (at random) "What object are we currently programming?" The answer is whichever object's name appears in the object name field. For this particular module, students should be working on (programming) the spring object.

Module 15: Create the Sprite (Image) for the Coin Object

OVERVIEW:

In this Module, students will: Create and draw a sprite for a new item, called Coin, using the tools in the built-in Image Editor.

Introductory Discussion : Many video games, especially classic arcade games, include events that score points for the player, along with functionality that keeps track of the player's score on the screen. The next addition to our game will be a coin item, which will do just that. When the player grabs a coin, his or her score will be increased, and the score will be displayed on screen. Ask students if they have ever seen coins in any of the games they play at home, and you are sure to get a few nods. Students will once again add animation in this drawing lesson.

Ask Students: We need a new sprite - what is the first thing we do?
Answer: choose Resources>Create Sprite

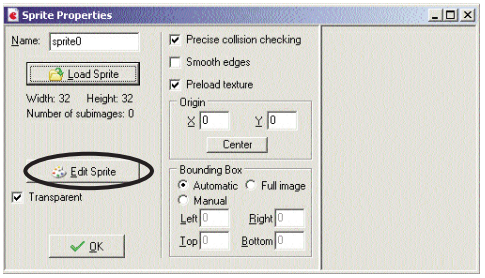
STEPS:

1. Choose Resources>Create Sprite (this will bring up the Sprite Properties window).
2. In the Sprite Properties window, click the Edit Sprite button (this will bring up the Sprite Editor window).

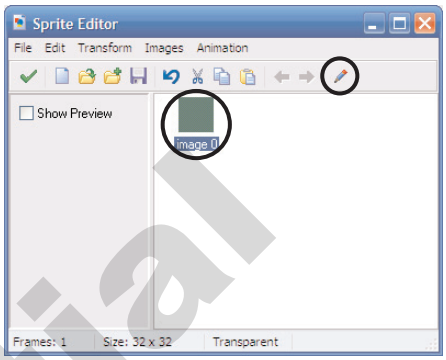
   Tell Students: Once again, we will draw and animate this sprite. First watch me and then follow along.
3. In the Sprite Editor window, click the Pencil icon, or double-click directly on the sprite image icon (this will bring up the Image Editor window).
4. In the Image Editor window, click on the magnifying glass (a few times) to zoom in on the art as you work (this does not affect the size of the image - it only affects the preview as you work).
5. On the left side of the Image Editor window, in the Image Editor toolbox, left-click to select the Draw an ellipse tool.
6. On the left side of the Image Editor window, in the Image Editor toolbox, select the third rectangle option on the bottom of the toolbox.
7. On the right side of the Image Editor window, in the Select Colors area, left-click to select a color with which to draw the coin (allow student to choose any color they like).
8. Position the cursor in the main art area, then press and hold the left mouse button, and with the left mouse button held down, drag the cursor to draw a circle as shown below:

Note: For this image and all the rest, just make sure the circle is always in the middle of the frame and not off to the side. Remember, you can always fix this by using the blue arrows on the top. Instruct students to do this as well.

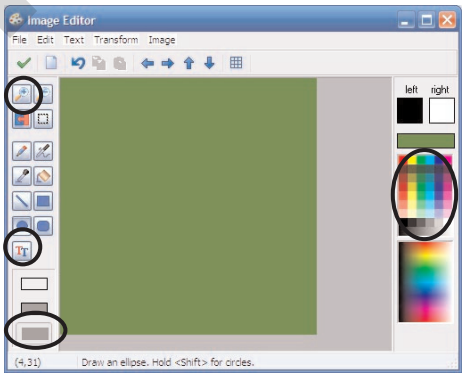
continued on following page



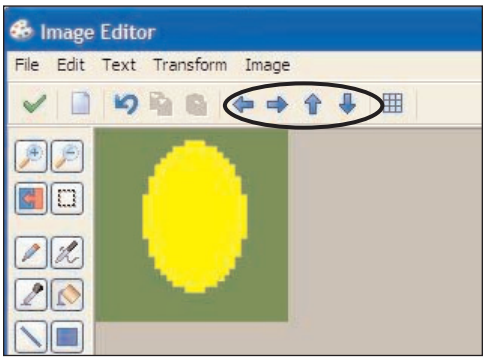
2. Click Edit Sprite



3. Click the Pencil icon or double-click directly on the sprite image icon



4,5,6,7. Click the magnifying glass to zoom in on the art as you work, left-click to select the Draw an ellipse tool, left-click to select the third rectangle option (lower left of window) left-click to select a color



8. You can always move the drawing by using the blue arrows on the top of the Image Editor

Module 15: Continued from previous page

9. In the upper left corner of the Image Editor window, click the green check mark (this will close the Image Editor window and return you to the Sprite Editor window).

   Tell Students: Instead of copying the image like we did with the spring, this time we will add a new empty frame.

10. In the Sprite Editor window, choose Edit>Add empty (this will create a blank frame in the Sprite Editor window).

11. In the Sprite Editor window, double-click image1 (the second image) to bring up the Image Editor window for that particular image.

    Tell Students: This time, we want to make the coin look like it is spinning. Here's a trick to make it look like that.

    Note: Draw the image below, a slightly squashed version of the first frame…

    Tell Students: As you can see, I made the circle a little bit thinner than last time.

12. In the upper left corner of the Image Editor window, click the green check mark (this will close the Image Editor window and return you to the Sprite Editor window).

    Tell Students: Let's add one more empty frame to start with. Where do I go to add a new empty frame? **Answer**: Edit > Add Empty

13. In the Sprite Editor window, choose Edit>Add empty (this will create a blank frame in the Sprite Editor window).

14. In the Sprite Editor window, double-click image2 (the third image) to bring up the Image Editor window for that particular image.

    Note: Draw the image below, an even thinner circle…

    Tell Students: This will be the thinnest circle. It's almost like looking at a coin on its side.

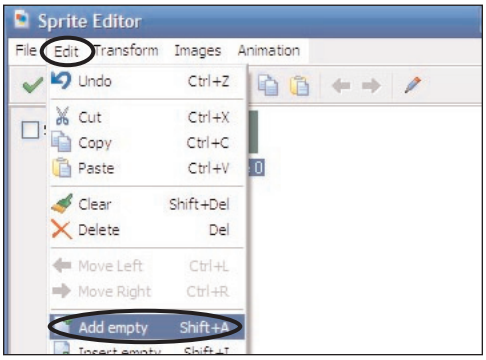
    Remind Students: Remember, you can use the blue arrows to move the image around. Try to keep it in the center.

15. In the upper left corner of the Image Editor window, click the green check mark (this will close the Image Editor window and return you to the Sprite Editor window).

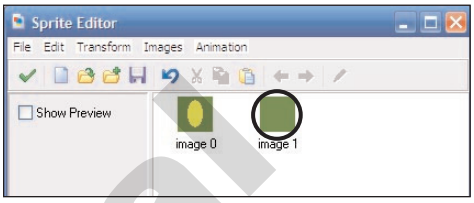
    Ask Students: Let's see what the animation looks like! Where do I go for that? **Answer**: Show preview

16. In the Sprite Editor window, click the Show Preview checkbox to preview the animation (preview speed can

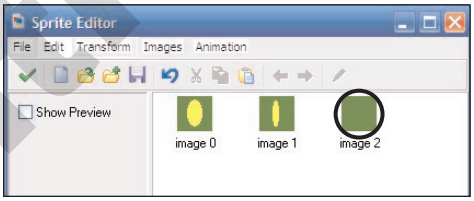
continued on following page



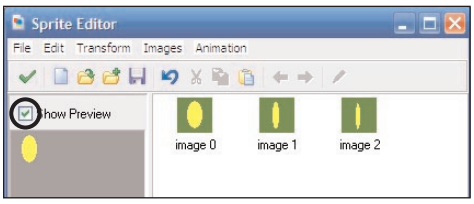
10. Choose Edit>Add Empty



11. Double-click image1



14. Double-click image2



16. Click the Show Preview checkbox

Module 15: Continued from previous page

be changed in the Speed option - though this is for preview purposes only. It does not necessarily represent the speed at which the animation will run in the game - that is programmed separately).

Tell Students: Great work! Now let's add an effect to make it look a lot smoother.

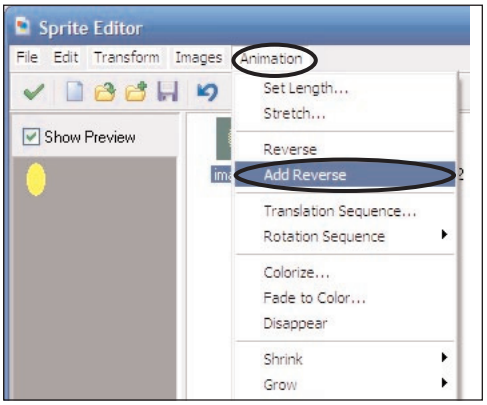
17. In the Sprite Editor window, choose Animation>Add Reverse (this will duplicate the existing frames and paste them into the Image Editor in reverse).

18. In the upper left corner of the Sprite Editor window, click the green check mark (this will close the Sprite Editor window and return you to the Sprite Properties window).

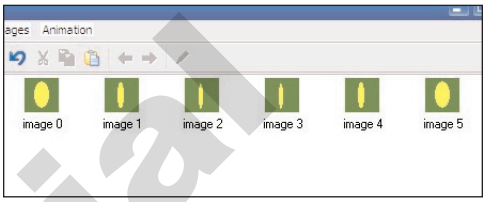
END

VOCABULARY:

Frame (Image): A single image in an animation sequence; when multiple frames are quickly displayed one after another, this creates animation.



17. Choose Animation>Add Reverse



After step 18, you should see six frames in the Sprite Editor window

VOCABULARY

Module 1: Gravity Example

Gravity: The force of attraction by which all things in the atmosphere are pulled toward the center of the earth.

Program: A series of instructions; a sequence of events.

Collision (in context of programming): A collision event is when any two objects touch each other in any way, at any speed.

Debug: To detect and remove programming errors.

Module 3: Draw Player Sprite

Sprite: An image used for an object or character in a game.

Object: Something that can be programmed (given instructions).

Avatar: Specifically, the sprite used as the player's character in a game.

Image Editor: Built-in drawing program included in the Game Maker software; used to create artwork for characters, backgrounds, and any other type of graphics for the game.

Graphic: A visual representation; an image.

Module 4: Program Player Character Movement

Gravity: The force of attraction by which all things in the atmosphere are pulled toward the center of the earth.

Program: A series of instructions; a sequence of events.

Event-driven programming: Programming that is based on an event occurring, and that event causing one or more actions to occur.

Object-Oriented Programming: Computer programming based on giving instructions to objects.

Cartesian Coordinate (Graphing) System: System invented by Rene Descartes to indicate position in two-dimensional space, based on x and y (horizontal and vertical) positions on a grid or graph.

x axis, y axis: Horizontal and vertical planes (respectively) of the Cartesian Coordinate System.

Event: An occurrence that triggers an action.

Action: An occurrence that takes place in response to an event.

Positive number: Any number greater than 0.

Negative number: Any number less than 0; marked with a minus symbol (-) before the number.

Horizontal: Pertaining to Left and right directions (x axis).

Vertical: Pertaining to up and down directions (y axis).

Module 5: Program Player Character Jumping

Gravity: The force of attraction by which all things in the atmosphere are pulled toward the center of the earth.

Vertical: Pertaining to up and down directions (y axis).

Horizontal: Pertaining to Left and right directions (x axis).

Module 6: Create the Sprite (Image) for the Grounder Object

Adversary: An opponent; enemy.

continued on following page

Vocabulary *Continued from previous page*

Animation: To bring to life; to give the illusion of movement.

Frame (Image): A single image in an animation sequence; when multiple frames are quickly displayed one after another, this creates animation.

Cell-by-cell (frame-by-frame) animation: The process of animating a series of drawings by running them quickly one after another, in the same physical position, so they appear to be as one moving image. This is the principle behind movie film and cartoon animation, among other uses for this technique.

Module 7: Create the Grounder Adversary Object

Object: Something that can be programmed (given instructions).

Module 9: Fix the Player's Jumping Functionality

Gravity: The force of attraction by which all things in the atmosphere are pulled toward the center of the earth.

Expression (in context of programming): A statement that uses variables, values, and equations; can be a statement tested by an event, or set in an action.

Module 10: Add Functionality for the Player to Defeat the Grounder

Collision (in context of programming): A collision event is when any two objects touch each other in any way, at any speed.

Self and Other: Refers to an instance of the current object being programmed; refers to an instance of the other object colliding with (or interacting with) the current object being programmed (respectively).

Vertical: Pertaining to up and down directions (y axis).

Module 11: Add Functionality for the Grounder to Defeat the Player

Collision (in context of programming): A collision event is when any two objects touch each other in any way, at any speed.

Self and Other: Refers to an instance of the current object being programmed; refers to an instance of the other object colliding with (or interacting with) the current object being programmed (respectively).

Transition: Change.

Module 13: Create the Sprite (image) for the Spring Object

Sprite: An image used for an object or character in a game.

Frame (Image): A single image in an animation sequence; when multiple frames are quickly displayed one after another, this creates animation.

Cell-by-cell (frame-by-frame) animation: The process of animating a series of drawings by running them quickly one after another, in the same physical position, so they appear to be as one moving image. This is the principle behind movie film and cartoon animation, among other uses for this technique.Module 14: Create the Spring Object and Program Player's Collision with Spring

Collision (in context of programming): A collision event is when any two objects touch each other in any way, at any speed.

Vertical: Pertaining to up and down directions (y axis).

Module 15: Create the Sprite (Image) for the Coin Object

Frame (Image): A single image in an animation sequence; when multiple

continued on following page

Vocabulary *Continued from previous page*

frames are quickly displayed one after another, this creates animation.

Module 12: Create the Coin Object and Program Player's Collision with Coin

Self and Other: Refers to an instance of the current object being programmed; refers to an instance of the other object colliding with (or interacting with) the current object being programmed (respectively).

Module 16: Adding Sound Effects for Collision with Spring

Sound Effect: A digital sound or noise, either synthesized or recorded, that is set to play on a specific occurrence; different than background music, which is a music track set to continuously play.

Loop: To return to the start after reaching the end; to play repeatedly.

Event: An occurrence that triggers an action.

Module 18: Adding Sound Effects for Collision with Coin

Sound Effect: A digital sound or noise, either synthesized or recorded, that is set to play on a specific occurrence; different than background music, which is a music track set to continuously play.

Module 21: Create the Lava Object

Lava: Molten, fluid rock that erupts from a volcano.

Animation: To bring to life; to give the illusion of movement.

Frame (Image): A single image in an animation sequence; when multiple frames are quickly displayed one after another, this creates animation.

Transparent (in context of sprites): Having an area of an image that is not visible; having a background color that will not be seen in game play.

Module 23: Create the Sprite (Image) for the Hopper Object

Animation: To bring to life; to give the illusion of movement.

Frame (Image): A single image in an animation sequence; when multiple frames are quickly displayed one after another, this creates animation.

Adversary: An opponent; enemy.

Module 24: Program Hopper Adversary Movement

Gravity: The force of attraction by which all things in the atmosphere are pulled toward the center of the earth.

Vertical: Pertaining to up and down directions (y axis).

Module 25: Add Functionality for the Player and Hopper to Defeat Each Other

Collision (in context of programming): A collision event is when any two objects touch each other in any way, at any speed.

Module 26: Setting Up the Second Level

Level (Room): The setting of the game; a game can have multiple levels (rooms).

Side-Scrolling: A common feature in video games that increases the size of the room (horizontally and/or vertically) without changing the size of the game window; allows a specified

continued on following page

Vocabulary *Continued from previous page*

object to be tracked around the room and travel beyond the edge of the original play field, similar to a video camera following a person or character.

Instance (in context of programming): A single copy of an object; In reality, you don't place objects in a room, you are instead placing copies (instances) of the object.

Width: Horizontal length.

Height: Vertical length.

Duplicate: To make a copy an existing resource (while preserving its details) which can then be changed.

Module 27: Create the Sprite (Image) for the Portal Object

Animation: To bring to life; to give the illusion of movement.

Frame (Image): A single image in an animation sequence; when multiple frames are quickly displayed one after another, this creates animation.

Rotation: The act of turning or revolving around a center point.

Degrees: A mathematical unit of measure to represent various angles.

Clockwise: Rotation in the same direction of the hands of a clock.

Counter-Clockwise: Rotation in the reverse direction of the hands of a clock.

Module 28: Program the Portal Object

Transition: Change.

Module 29: Create the Sprite (Image) for the Cloud Object

Frame (Image): A single image in an animation sequence; when multiple frames are quickly displayed one after another, this creates animation.

Animation: To bring to life; to give the illusion of movement.

Obstacle: Something that stands in the way of accomplishing a task or attaining a goal; an obstacle can be physical or in another form.

Precise Collision Checking (in context of sprites): If turned on, collision events for the associated object will happen specific to the shape (pixels) of the sprite; if turned off, collisions will not be specific to the shape of the sprite, and will instead correspond with an invisible box (bounding box) made around the sprite; for example, turning this off for a sprite with an image of a circle, will turn its actual area of collision to a square (not specific to the shape of the sprite).

Module 30: Program the Cloud Object

Collision (in context of programming): A collision event is when any two objects touch each other in any way, at any speed.

Subimage: Refers to a specific frame of animation for a sprite.

Duplicate: To make a copy an existing resource (while preserving its details) which can then be changed.

Module 31: Create the Win Screen

Goal: Something desired and strived for by an individual or group.

Background: A non-interactive image usually set behind the objects of a room; applied directly to a room (not to an object).

continued on following page

Vocabulary *Continued from previous page*

Module 32: Create the Intro Screen

Background: A non-interactive image usually set behind the objects of a room; applied directly to a room (not to an object).

Font: Style and size of text.

Module 33: Add a High Score Board

Goal: Something desired and strived for by an individual or group.

Module 34: Create an Executable File

Executable File: PC format file that can be run on any PC, regardless of the software used to create it.

Stand Alone Game: Executable file; can be run without specific software.

Vocabulary - Alphabetical List:

Action: An occurrence that takes place in response to an event.

Adversary: An opponent; enemy.

Animation: To bring to life; to give the illusion of movement.

Avatar: Specifically, the sprite used as the player's character in a game.

Background: A non-interactive image usually set behind the objects of a room; applied directly to a room (not to an object).

Cartesian Coordinate (Graphing) System: System invented by Rene Descartes to indicate position in two-dimensional space, based on x and y (horizontal and vertical) positions on a grid or graph.

Cell-by-cell (frame-by-frame) animation: The process of animating a series of drawings by running them quickly one after another, in the same physical position, so they appear to be as one moving image. This is the principle behind movie film and cartoon animation, among other uses for this technique.

Clockwise: Rotation in the same direction of the hands of a clock.

Collision (in context of programming): A collision event is when any two objects touch each other in any way, at any speed.

Counter-Clockwise: Rotation in the reverse direction of the hands of a clock.

Debug: To detect and remove programming errors.

Degrees: A mathematical unit of measure to represent various angles.

Duplicate: To make a copy an existing resource (while preserving its details) which can then be changed.

Event-driven programming: Programming that is based on an event occurring, and that event causing one or more actions to occur.

Event: An occurrence that triggers an action.

Executable File: PC format file that can be run on any PC, regardless of the software used to create it.

Expression (in context of programming): A statement that uses variables, values, and equations; can be a statement tested by an event, or set in an action.

Font: Style and size of text.

Frame (Image): A single image in an animation sequence; when multiple

continued on following page

Vocabulary *Continued from previous page*

frames are quickly displayed one after another, this creates animation.

Goal: Something desired and strived for by an individual or group.

Graphic: A visual representation; an image.

Gravity: The force of attraction by which all things in the atmosphere are pulled toward the center of the earth.

Height: Vertical length.

Horizontal: Pertaining to Left and right directions (x axis).

Image Editor: Built-in drawing program included in the Game Maker software; used to create artwork for characters, backgrounds, and any other type of graphics for the game.

Instance (in context of programming): A single copy of an object; In reality, you don't place objects in a room, you are instead placing copies (instances) of the object.

Lava: Molten, fluid rock that erupts from a volcano.

Level (Room): The setting of the game; a game can have multiple levels (rooms).

Loop: To return to the start after reaching the end; to play repeatedly.

Negative number: Any number less than 0; marked with a minus symbol (-) before the number.

Object-Oriented Programming: Computer programming based on giving instructions to objects.

Object: Something that can be programmed (given instructions).

Obstacle: Something that stands in the way of accomplishing a task or attaining a goal; an obstacle can be physical or in another form.

Positive number: Any number greater than 0.

Precise Collision Checking (in context of sprites): If turned on, collision events for the associated object will happen specific to the shape (pixels) of the sprite; if turned off, collisions will not be specific to the shape of the sprite, and will instead correspond with an invisible box (bounding box) made around the sprite; for example, turning this off for a sprite with an image of a circle, will turn its actual area of collision to a square (not specific to the shape of the sprite).

Program: A series of instructions; a sequence of events.

Rotation: The act of turning or revolving around a center point.

Self and Other: Refers to an instance of the current object being programmed; refers to an instance of the other object colliding with (or interacting with) the current object being programmed (respectively).

Side-Scrolling: A common feature in video games that increases the size of the room (horizontally and/or vertically) without changing the size of the game window; allows a specified object to be tracked around the room and travel beyond the edge of the original play field, similar to a video camera following a person or character.

Sound Effect: A digital sound or noise, either synthesized or recorded, that is set to play on a specific occurrence; different than background music, which is a music track set to continuously play.

Sprite: An image used for an object or character in a game.

Stand Alone Game: Executable file; can be run without specific software.

Subimage: Refers to a specific frame of animation for a sprite.

Transition: Change.

continued on following page

Vocabulary *Continued from previous page*

Transparent (in context of sprites): Having an area of an image that is not visible; having a background color that will not be seen in game play.

Vertical: Pertaining to up and down directions (y axis).

Width: Horizontal length.

X axis, Y axis: Horizontal and vertical planes (respectively) of the Cartesian Coordinate System.

Game Design Document: Planning Your Game

Your Name: _____

Game Title: _____

(Use the back of this page or more paper if you need more room)

Player Character Description and Capabilities:

Objectives/Goals:

How Player Wins or Goes to Next Level:

How Player Loses:

continued on following page

GAME DESIGN DOCUMENT: *Continued from previous page*

Obstacles:

Opponents/Adversaries/Non-Player Characters:

Rewards and Power-Ups:

Setting (Game World):

Special Graphics: