Kodu Curriculum: Keyboard and Mouse

Table of Contents

Kodu Curriculum: Keyboard and Mouse

Welcome to keyboard version of the Curriculum for Kodu (keyboard version)! What follows is a resource for using Kodu as part of a class, after school program, or summer camp. This provides a basic scope and sequence intended to introduce instructors and students to the Kodu application, and basic computer programming concepts, and it suggests ways that Kodu might be used to complement traditional curriculum.

The main goal of Kodu is to expose students to computer programming in a fun environment, while getting them excited about potential careers in computer science by allowing them to create their own games. This makes them producers of their own media rather than just consumers of it. More specifically, after using Kodu, students will:

- ✓ Better understand the steps involved in creating a computer program
- ✓ Improve problem solving skills, and foster problem solving practices
- ✓ Follow online and offline directions more fluidly
- ✓ Learn to compose stories in an alternative format and through varying mediums
- ✓ Implicitly practice math through branching and scoring
- ✓ Develop more positive attitudes towards computer programming
- ✓ Create increasingly complex games thus showing a deeper understanding for complex coding sequences
- ✓ Show evidence of perspective taking and empathy in game play
- ✓ Collaboratively work to create innovative solutions

Kodu Description

The user interface is the foundation of working with Kodu. The language is simple and entirely icon-based. Programs are composed of pages, which are broken down into rules, which are further divided into conditions and actions. The Kodu language is designed specifically for game development and provides specialized primitives (the nouns, adjectives, and verbs of the language) derived from gaming scenarios. Programs are expressed in physical terms, using concepts like vision, hearing, and time to control character behavior. Kodu can express advanced game design concepts in a simple, direct, and intuitive manner.

Teaching with Kodu

We suggest that the best way to teach Kodu is through studio pedagogy with mini-lessons interspersed throughout as a means to move the students along in their discovery of the tools while enacting collaborative, problem solving practices. Note, however, the environment in which Kodu is presented, along with the pedagogical inclinations of the facilitators, might require a looser or tighter structure. In short, the instructor should take what track she or he feels best suits the class and his or her teaching style. Our goal is to help address the many ways that students respond to technology—some like to be directed in their advancement and others tend to forge ahead, experimenting through trial and error. There is room for both types of student in this curriculum.

About the Developers

Kodu was developed by a team of research programmers at Microsoft who are passionate about kids having fun and being challenged as they learn how to program.

For More Information

Visit http://research.microsoft.com/en-us/projects/kodu/ for more information about Kodu and its developers.

Visit the Kodu blog to see what others say about Kodu or post your thoughts about the program.

There are some routines and pedagogical practices we suggest to help facilitate the class and enrich the learning experience of the students. First, while explicit instruction about the tools and the criteria for final projects is important, you should also let students playfully experiment with the tools. This means that they will make mistakes requiring active problem solving.

Problem Solving Culture

To help facilitate deliberate mistake-making, students should be continually reminded that they, as a collective, hold the keys to problem solving. “Three-before-me” is a technique used to assist students in collaborative problem solving conversations –simply put, if students can’t figure out an issue they ask three other students before coming to the instructor. Another method is writing problems on a “Post It” and posting them to a collective space where students share and answer each other, much like a digital discussion board. As students answer questions with another electronic “Post-It,” the questions move to solved side of the board.

A Culture of Critique

Studio pedagogy allows a space for students to discuss their work and receive feedback—this process is integrated throughout the game creation process rather than sequestering the activity at the end of a draft, as in the traditional writing process, or at the end of product creation. There are several ways of establishing this culture the class. One method is to have students work in small groups, taking turns to discuss each project’s intentions, challenges, and successes while others in the group give warm and constructive feedback. Often, this needs to be modeled. Another implementation strategy might include regular whole class exhibitions during which students peruse the games at various stages in development and ask questions of other students’ projects. Again, this strategy would likely need to be modeled, perhaps through a fishbowl methodology.

We suggest that you use the studio format at least once a week if you meet daily. We also suggest that the class close with a discussion that outlines the successes and challenges from that day's session—this can be directly related to Kodu issues or to more general in-class processes and classroom climate issues.

Reflective Practice

It is also suggested that you and your students keep a journal about their work with Kodu. It is best if journaling becomes part of class ritual, perhaps at the end of each session. Guiding questions help facilitate this process. These same journals might also house student planning of games once they start working on their work game projects.

Exhibition of Student Work

We advise that you end your Kodu unit with an exhibition of student work. Students can introduce their games to an audience of peers, parents and other adults. For added interest, consider coming up with the components of a successful game with the students. This can be used as the criteria for evaluating the games.

Working with the Keyboard and Mouse

Kodu was originally designed with the user playing and editing worlds using an Xbox controller. Being aware that controllers may be cost prohibitive for many schools, community centers and households, Kodu developers created the keyboard version of the software. However, most of the game worlds within Kodu still require the controller to play them since they have not yet been converted. This can pose some issues when users start working with the program. At the same time, it also forces users to make the game worlds work for them by editing the code.

There are two primary methods to make an avatar or player-controlled characters move in the keyboard version.

Method 1 uses more generalized direction through the arrow keys. In the code structure, simply write in the code area WHEN: keyboard DO: Move. The up arrow represents forward movement, and the left and right arrows represent left and right turning respectively. This method does not rely then on the compass and north, south, east and west.

Method 2 is via the arrow keys being programmed to a particular direction (North/Up, South/Down, East/Left and West/Right). Using this method, however, can present challenges since North may not be Up (or South may not be down, etc) depending upon how the landscape has been created in relation to where the players or characters are placed. This is can be confusing when playing a game, but again, the worlds and characters can be reprogrammed and redesigned to accommodate a keyboard and mouse. When using this method, the compass indicator is an essential tool when navigating worlds using the arrow keys. Some worlds do not have the compass on the interface which indicates direction; in order to display the compass, enter Edit mode within a game world by pressing Escape and selecting the final icon in the list (represented as a wrench and mountain landscape). Then, arrow down to Show Compass and make sure it is selected.

Kodu Curriculum Scope and Sequence

Session 1 Navigating, Intro to Programming Concepts, Adding Objects

Session 2 Creating a Landscape

Session 3 Using Controller to Move Characters, Create Paths, and Set Behaviors

Session 4 Making clones and Creatables

Session 5 Starting Unique Stories and Characters

Session 6 Strategy, Mood, and Tone

Session 7 Changing Behaviors Using Pages, Establishing and Shifting Perspectives

Session 8 Power Ups, Health, Timer

Session 9 More on Scoring--Basics to Communication

Session 1: Navigating, Intro to Programming Concepts, Adding Objects

When finished, students will be able to:

- Navigate the Kodu macro environment using a keyboard and mouse
- Understand the foundational principles of programming
- Access the programming mode of Kodu, potentially adjusting simple code for a specific purpose

Survey—get to know each other through the survey

Before actually venturing into Kodu, please have your students take the Kodu survey. The survey will act as a springboard for a conversation that allows you to get to know your students better. The data gathered from the survey helps the developers of Kodu, and it also supplies teachers with some useful information about the types of technology practices in which their kids engage.

Exercise in Programming an Unsuspecting "Avatar"

An in-class exercise is handy to prime students with some idea of what it is like to program games in Kodu.

Materials:

- 3 red apples (balls or some other colored object will do)
- 2 green apples
- 1 bag
- 1 or 2 blind folds

Inform the class that in Kodu they will need to:

- Select a character(s) and make it behave and react in certain ways,
- Build an environment in which the characters operate
- Fill that environment with objects that the characters either interact with or not
- Create rules and reactions that allow the characters to function in the environment

Ask for a student volunteer to act as a Kodu character. Have the student sit in a chair in the middle of the room blindfolded. Then ask the other students how the room should be arranged to create a Kodu environment. Tell them that you have 5 objects to place in the environment (3 red apples and 2 green). Place them in different spots in the room. Inform them that we need to direct our Kodu character to find and pick up the apples. Note that every action must be outlined and described in detail.

To help organize their “commands,” inform students that the character only listens to you, so they need to filter the commands. Ask for clarification and further detail if something is vague or difficult to enact. Depending on how energetic your students are, you may need to call on students to help with classroom management.

Keep asking how, when, and where question. Feel free to remove the blindfold from the student if he or she is prompted to see (this type of direction is referred to as a “command” in Kodu).

During the exercise consider the following options that Kodo can respond to:

- Forward and backward
- North, South, East, West
- Fast and slow
- Jump and turn
- Bump up against things
- It can be programmed to move toward and away from certain objects
- Kodu can see, hear, and distinguish colors
- Kodu can express love, anger, sadness, and craziness
- Kodu can say things through text as well
- Kodu can even wait at places
- Kodo can move between places within a certain amount of time

Things to Remember: All of these actions and reactions need to be programmed by the user. So, as you take commands from the class, be sure to continually ask them how, when, why, and where in order to get increasingly detailed about the action.

A typical scenario might be:

Teacher: We want Kodu to do something that involves these apples. Let's try to just move our Kodu to the apple and put it in his bag. What would be the first thing we would tell it to do?

Student: Move to the apple?

Teacher: How? Does he walk?

Student: Yes, he walks?

Teacher: Fast or slow?

Student: Fast.

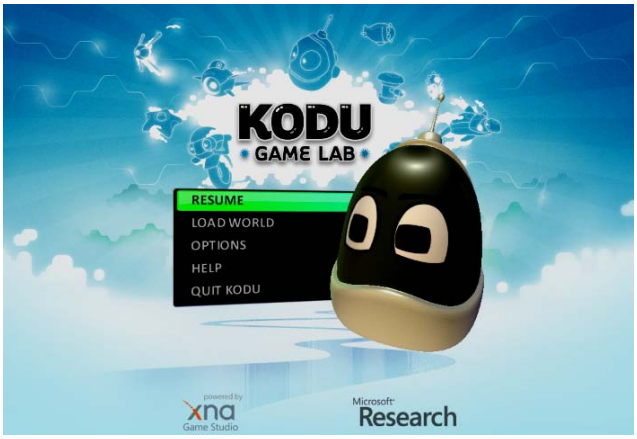
Teacher: How does he know when? What prompts him to move?

As the scenario transpires, it might be helpful to write the commands on the board so the class can easily recall them. Once a coherent set of commands are established, put your Kodu in motion.

Introducing Navigation and Kodu Main Page

Before having the students log onto Kodu show them the basic structure of the program through the **Main Page**. It has five options that the user either arrows through or selects by left-clinking with or by pressing **Enter**.

1. Resume



Opens whatever game or level that was edited last by the user. If the user opens this window, the last game that was programmed or opened will appear.

To back out of this window, click the **House** icon and then select **Exit to Main Menu**.

2. Load World

Opens a series of games or levels from which the user can select. The levels can be perused by using either the arrow keys or clicking the on screen arrows with the mouse.

To go back to the **Main Menu**, users press **Escape**.

3. Options

Provides the user with the ability to change some broad level features of the game. To navigate this list, users either use the arrow keys or hover the mouse cursor over the green arrows on the screen. **Escape** will bring the user back to the **Main Menu**.

4. Help

Offers a quick explanation of Kodu. Users press Escape to exit this menu.

5. Quit

Allows users to leave the program.

Eating Apples Intro

To echo the activity of moving and coding figures in the room, turn to **Activity 1 Eating Apples** level in Kodu. Navigate to the level during class, and tell the students to take note of everything that is happening on the screen. Select play to run the program, and ask what they notice.

- Kodu eats the apples.
- Kodu moves from apple to apple.
- Kodu does not eat the green apple.
- The tree produces more apples and they pile up behind Kodu.

Now, show the students the program code that is running this set of behaviors—Kodu and the tree. The code can be read by pressing **Esc** when the game is launched.

Next, select the Kodu icon from the toolbar and hover the cursor over the object (Kodu or tree) of which you want to read the code. The object will brighten. Clicking the right mouse while holding it down will allow the user to move the object, while clicking the left mouse will bring up a slew of options for the user to choose from.

Program will bring up the code menu. Read the code together and ask if they see these behaviors in the code for both the Kodu and the tree. After reading the code, edit the world while asking students how the code could change for the set of behaviors.

- Add an apple. [This is produced by choosing Kodu from the tool bar with the mouse, and then clicking on the landscape where you want the apple to go. The object selection ring will appear, and select the apple tile.]
- Make one apple in the sequence green; make sure Kodu doesn't eat it. [Click on or hover the cursor over one of the apples in the circle of apples and the color palette at the top will appear. Using the arrow keys, make one of the apples green.]
- As a challenge, adjust Kodu's behavior so it turns to look for more apples when he sees none. [There are various levels to this command depending on how far you want to program Kodu. The following sequence should allow Kodu to continuously search for apples. Simply adding the code to turn at the bottom of the programming sequence will not work. Follow this sequence to get Kodu to turn and look for more apples.

  (Line 1) When: see none apples Do: move turn quickly.

  (Line 2) When: see apple red Do: move toward.

  (Line 3) When: bump, apple red Do: eat.

  (Line 4) When: see none apple red Do: move turn quickly

  (Line 5) When: see apple red, Do: move toward quickly.

This activity is very worthwhile for the students since it models the trial and error that is often involved with programming. After modeling the strategies, searching for code, and assembly process, have the students practice with **Activity 1 Eating Apples**.

Reflection

After the students work through **Activity 1 Eating Apples**, tell the kids to save their projects and turn off their monitors. It is time for reflection. Ask the students the following questions:

- What was challenging?
- What was easy?
- What was a success?
- What did they learn or what did they already know?
- What do they want to learn next?

What You Can Expect from the Student Activity

Free time is essential for students to become familiar with the tool and some general programming concepts. You will find that most of the students will work through the list that you provide and then explore the various features on their own—adding more characters and objects and giving them behaviors.

Some students will even work with the landscape. Others will start playing games and not even look at the To Do list. What you allow is up to you, but it is always good to regroup to share their experiences and thoughts.

Also, if any students are working in tools that you will be covering in the coming lessons, then consider having those that students teach other students how to use that feature.

Student Sheet Activity 1: Eating Apples

Objectives: Add object, change color, select object, create text based on event, change movement based on non-event

Directions: We just went over some of these steps as a group in Activity 1 Eating Apples. It is your turn to try it out. As you complete each of the following, CHECK IT OFF YOUR LIST.

Once you are done, check in with your advisor and have them look at what you have completed. The underlined words are meant to be clues to finishing your code.

To Do Check List

- ❑ Add an apple to the sequence of apples—make it blue.
- ❑ Select an existing apple and change its color to blue.
- ❑ Make Kodu say (found under actions) "I am hungry" when he bumps the blue apple.
- ❑ If apples are piling up under the tree after the Kodu sequence is done, make Kodu turn around so he can see the apples and eat them.
- ❑ If the apples are not piling up, make an apple behind the tree that the Kodu is attracted to it. This should make the apples start piling up. When you have programmed it so the apples are piling up, then code Kodu to turn around when he sees none and look for the other apples.

Once you complete the to do list, try adding other objects, adjusting Kodu's behaviors, and changing the environments. Please save your work. Ask for help if you need it.

Challenge Activity

As a challenge activity, go to **Tutorial 01 v2** and do what the Kodu asks. Also, see if you can reprogram the castle to behave in a different way once it is bumped, and try to figure out how the camera can follow Kodu on its trek to the castle. Another apple eating scenario is presented to you in **Technique: Eating only certain apples v07.** Take a look at it and see if you can reprogram according to the entry screen.

Session 2: Creating a Landscape

When finished, students will be able to:

- Change and create a Kodu environment
- Use tiles for setting and the development of tone and mood of game world

Creating and changing the Kodu landscape is one of the most interesting initial activities for early users of Kodu. They often spend hours changing and designing intricate landscapes in which their games and animations operate. Landscapes set the tone for actions that follow. For these reasons, we devote a good amount of time to the landscape features and student experimentation. The intentions of the activities are for the kids to begin building worlds from the ground up (literally) and thinking creatively about their use of tiles and how to assemble them for a particular effect.

Whole Class Instruction

Using a computer hooked up to a projector, ask for student volunteers to operate several of the landscape features—some of them will have likely used the landscape feature during the previous session. Opening an empty world ask students to demonstrate how to add and delete land, as well as give it color and texture. Ask them to verbalize what they are doing so the class can hear their thought process and instructions. As you move into talking about mood and tone, be sure to have students try to define these terms. Even students in middle school might have trouble with what tone and mood mean, so be explicit in your instruction and repeatedly come back to them as the class progresses through the lessons.

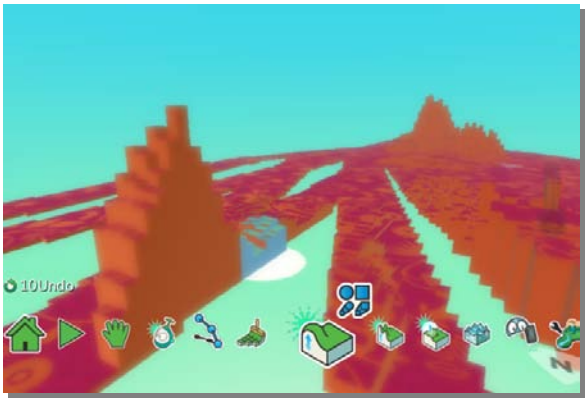
- Add/delete land

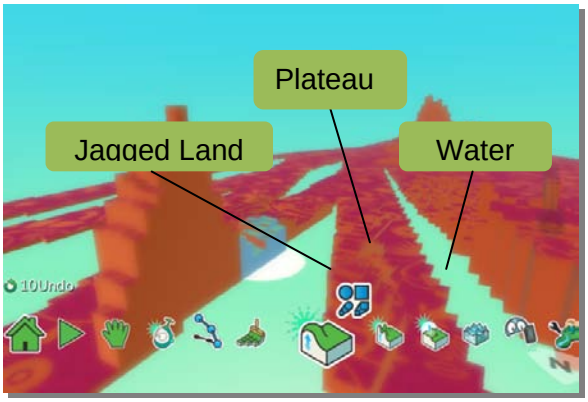
Choose green paintbrush in the toolbar. Select the appropriate landscape material by clicking on the materials icon to the upper left of the paint brush. Then, use the arrows to make a selection. Once one is found that seems appropriate, click it. To apply land, press and hold down the left mouse button while moving the mouse to “paint” the landscape. By holding down the right mouse button, and moving the mouse, the user can erase land formations.

- **Choosing an appropriate edging**—Choose the green paintbrush in the toolbar and the appropriate paintbrush by clicking on the geometric shapes icon to the upper right of the icon. Arrow through the choices and select the desired edge. The square brush will give

users a flat edge while the circle will supply a more rounded edge. The extended square and circle icons allow users to draw stretches of land with a clearer edge, and the magic wand icon supplies an easy way to change the color and texture of a landscape en masse with ease.

- **Create hills and valleys; use the smoothing feature**—Once a landscape is created, land can be raised and lowered. There are a number of affects to choose from when lowering and raising ground. The first icon in the series shown below allows for the land to be raised or lowered as a collective based on where the paintbrush is located. The second icon in the series raises and lowers more jaggedly, and the third icon allows the user to flatten out a surface, thus creating a plateau.





- **Create an island or a lake**—The water icon in the toolbar allows users to add lakes, seas and oceans to their worlds. In order to add these environments, the designer of the world must first create a landmass on which the water rests. Notice in the above screenshots how the water did not fill the spaces in which land was not drawn. When the water icon is clicked, users are given options for the color of the water. These can be perused using the arrows and the mouse.

- **Adjusting settings for water and sky**—The settings icon at the very end of the toolbar provides some of the most powerful features in Kodu and impact both the movement in and look of the world. Thus, there are some settings that influence tone and mood a great deal. Here are a few settings that have an impact on mood and tone, as well as the general creation of the world.

- Glass Walls is the default setting for all game worlds and acts to contain the game play to the drawn environment. Without the walls, characters can careen into the abyss outside the game world.

- Show compass allows the user to understand which direction they are moving. This is particularly important when users start moving characters and using arrow keys to drive movement.

- Wave Height adjusts the size of the waves in the game world

- Water Strength changes the level of distortion of the water.
- Sky provides the user with varying atmospheric types.
- Lighting establishes the amount and type of light is cast in your world.

After you have covered each of these areas with your students acting as drivers in the creation of the game world, have them experiment with the tools that make a landscape, as well as create tone and mood. The contrast in the students' projects will supply material for an interesting dialogue about setting and its role in a game.

Changing the Sight Line

While students are building their worlds, they may find it difficult to see what they are actually crafting depending upon the angle in which they view their worlds. To help navigate their worlds as they are building them, show them the **Green Hand** on the toolbar. When the mouse is left clicked, the cursor allows the user to push the landscape forward, backward, and to the sides. Additionally, the center mouse scroll allows the user to zoom in and out, and if the user holds down the left and right mouse buttons at the same time, a game world's sight line can be shifted vertically and horizontally.

Reflection

During the last 20 minutes of class, have the students share their worlds. Ask them to share with each other what they find interesting about one another's worlds. If they finish this activity early, tell the students to feel free to explore the other worlds that already exist in Kodu.

What You Can Expect from the Student Activity

The landscape tools are particularly riveting for the students. Some students will spend hours building and revising their environments to create a specific affect. In fact some of the kids will become so engrossed in their world, they will put their efforts in landscaping as opposed to character development, game strategy, plots and usability. It is important to remember that people tend to compose and create in different ways. Consider writing; sometimes the intent of the author is for personal satisfaction over audience interest. This may also hold true for the students while working in Kodu. If you find that this is the case, it would be beneficial to discuss these approaches to composition and intent as a class once you have entered into discussions of characterization and plot.

Subject Area Link—Geography

If you think it is appropriate, consider making an instructional link to geography. Amazing landscapes can be designed and demonstrated with Kodu.

Depending on the grade of the student, you might ask them to illustrate different geographical terms: hills, valleys, mountains, peninsula, cliff, island, bay, isthmus, volcano, plateau, channel, etc.

Kodu can also illustrate changes to landscapes: consider for instance, illustrating erosion, landmass changes due to earthquakes and volcanoes, glacier activity, etc.

An interesting exercise for students may be to create a topographic map of a mountain(s), river, state, province or country, perhaps even illustrating the same land mass pre and post a geological event.

Student Sheet Activity 2: Making Landscapes

Objectives: Create land with texture, add water, trees, rocks, etc.

Directions: We just went over the different tools for creating a landscape. It is your turn to try it out. As you complete each of the following, check it off your list.

Check in with your instructor once you are done.

To Do Checklist:

- ❑ Go to Empty World.
- ❑ Create a landmass with:
  - At least two types of materials
  - Create rolling hills, mountains (with a white peak), and valleys
- ❑ Make an island or two off the coast of your land
- ❑ Add water as either a river, lake, ocean or all of the above
- ❑ Create a magical forest somewhere in your landscape (You can define magical in any way you want. There are number of objects to choose from: trees, rocks, stars, coins, etc.)
- ❑ Create storm clouds over one part of your landscape.

Challenge Level

After you have created your world, see if you can find where to change the mood and tone of the game. Specifically, try changing some of the settings. Investigate the following settings and note how they change the meaning of your world:

- ❑ Wave height
- ❑ Water strength
- ❑ Sky
- ❑ Lighting
- ❑ Breeze

How do adjustments to the setting change the feeling of your world? Come prepared to share your world with your classmates.

Session 3: Using Controller to Move Characters, Create Paths, and Set Behaviors

When finished, students will be able to:

- Use the keyboard to move characters in a game world
- Create paths on which characters will move
- Give objects behaviors

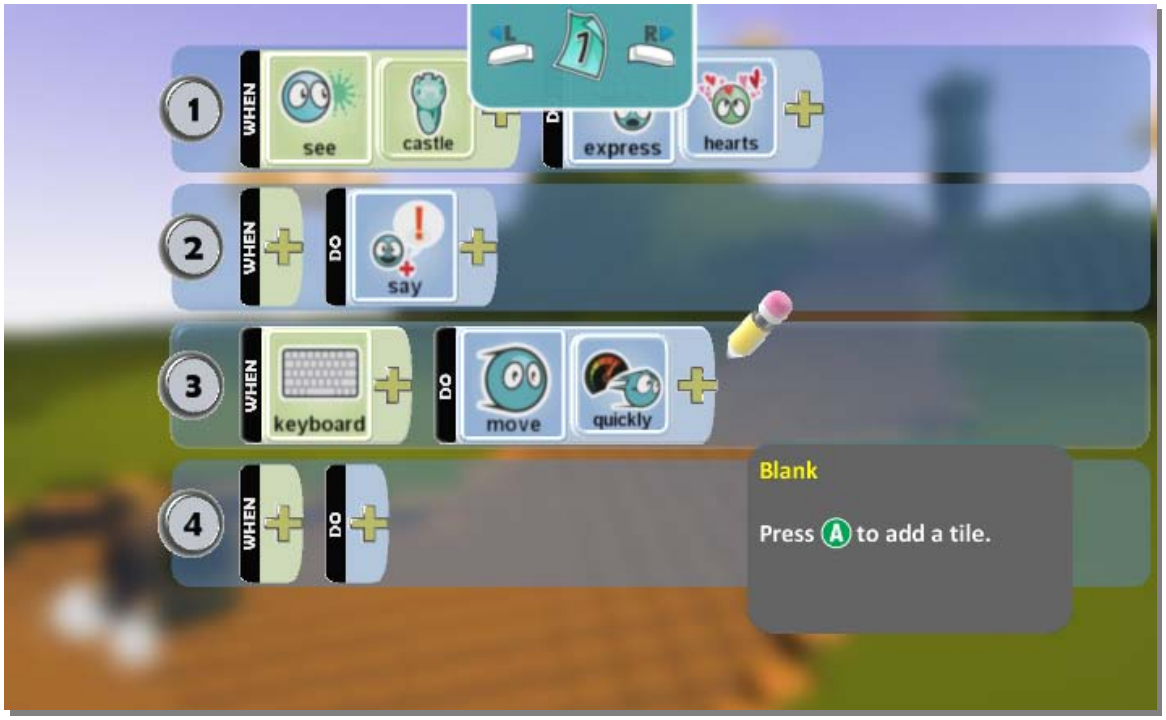
Thus far in the lesson plans, students have not been asked to program characters to make them move (except that is in the **Challenge Level** in Session 1 using **Tutorial 01 v3**). This is not to say that some students have not already figured out how to do this. In all likelihood some have, while others have been content to follow the script of the student activities. In any case, it may be productive to have the students once again lead each other in these activities. If students have worked ahead, have them demonstrate to others how to program user guided movement, create paths and give objects behaviors.

Whole Group Instruction

For the students who did **Tutorial 01 v3**, this would be a good time for them to share how they solved the gaming puzzle. Using the projector, ask for a student volunteer to demonstrate how they made Kodu go to the castle. As a reminder, after opening the game world of **Tutorial 01 v3**, press **Esc** and click on the Kodu icon in the toolbar. Put the cursor over the Kodu avatar and right click the mouse. Choose **Program**. The current code is below. There are two ways primary ways to make the Kodu move to the castle. In either case, the user must enter edit mode, choose the Kodu icon, and edit its behavior. The Kodu can be programmed either to move as if through animation or through user control via the arrow keys.

To animate Kodu to move to the castle, add the following code to the sequence.

To control the movements of Kodu, two programming sequences can be used.
The first uses generalized directions of **forward**, **right**, and **left**.



A second method for moving Kodu includes using the directions North, South, East and West.

If the student demonstrates one of the above methods, be sure to suggest an alternative method to show that the problem can be solved in multiple ways. Also be sure to have the students read the code that precedes the new line that is entered. It is good for the students to practice reading the syntax of the code.

Next go to **Idyll KB**. This game has a number of components from the lesson. Instead of playing the game right away, have the students read the code either in small groups or collectively. If they work in small groups and report back to the larger group, be specific about the code you want them to look at: Kodu, the castles, the blimp, and tree.

If working with the entire class, start with the Kodu—the code is fairly straight forward. While you can go to Page 2 and read the code, pages will be covered in the next lesson. Now, run the game. Ask the students what is happening in the game that they didn't see in the code. They will likely notice that the castles disappear and create wisps. They will also notice the blimp moving. And, they may also notice that points are accumulating—this is programmed through the tree. Go back and take a look at the code for each of these components. Of most interest to us for this lesson are the castle and the blimp. Scoring will be covered in Lesson 6. A wisp is also coded into the game as a creatable, but this feature will be covered in Lesson 4. Read the code for the castle. Again, it is straight forward—when Kodu bumps the castle it releases a wisp and it blows up. Next, read the code of the blimp—there is only a DO statement, no WHEN. Ask the students what this means.

Paths

The idea of a path will most likely be new to the students. Show them the height of the path can be raised and lowered by highlighting a node, left clicking and then choosing **Change Height**. Also note that the color and type of the path need to be specified in order for the blimp to move along the path.

While looking at **Idyll KB**, ask them how they would create a second path for a hot air balloon. This is done by choosing the **Path** icon from the main toolbar. Once the **Path** icon is clicked, use the left mouse button to add nodes, which act as angle points to connect the path sequence. Simply left click every time a node is needed in the path. Right clicking on a node brings up more options—allowing for the deletion of a node, the adding of more nodes to the sequence, the changing of height of a path, and the changing of the path to a particular type of path, e.g. a road, a wall, a platform, etc.

Reflection

Toward the end of class, ask the students to save and shut off their screens for a few minutes to discuss what they found during their lesson. See if they had similar problems and if solutions were found. Discuss how they solved problems—sometimes this entails trial and error, other times this includes looking at code in the other programs. The activity is meant for students to share strategies.

Subject Area Link – Science

The path feature and the coding tiles Move /Toward and Fast/Slow in Kodu could be particularly useful in animating biological and physical happenings.

Consider, for instance, animating the tribulations of fish or turtle during their migratory paths from sea (dodging debris and fishing vessels) to locks and fish ladders to natural predators. Concepts like over fishing might easily be put into a game format, too.

From a physical science perspective, Kodu might animate stellar patterns, trajectories or atom formation. Animating these concepts can help students to retain concepts with greater depth and complexity.

Student Sheet Activity 3: Object Behavior and Paths

Objectives: Students move characters using the keyboard and on their own, program object behavior, create a path that a character/object will follow

Directions: We just went over the tools for making characters move. It is your turn to try it out.

As you complete each of the following, check it off your list. Check in with your instructor once you are done or if you are having trouble.

To Do Check List:

- ❑ Go to the landscape you created during the last session or to an existing game in Kodu. For simplicity's sake, choose a world that has land of some sort.
- ❑ Create a character that the user controls with the controller.
- ❑ Create a second character that has automated movement.
- ❑ Create a path on which a third character moves.
- ❑ Create an object that does something either when it is bumped, sees or told to do something using the controller or when it is programmed to do something automatically through a DO statement.

Challenge Activity

After you have finished your TO DO for Activity 3, try your hand at **Tutorial 02**. See if you can figure out how to program the motorcycle to move and fire using the keyboard and mouse according to the instructions.

Also take a look at **3D Flare Paths** to see how to make 3D representation and interesting graphic affect.

Check out the action game **Rock Fight v09** for an example of elevated paths. If you want to play the game, you will need to convert it to a keyboard version of the game. Be sure to change the entry screen for the user, directing them on what keys to use when playing the game. You can change this screen by going to the Wrench icon in the main toolbar and changing the **Start Game With World Description**. Also, when playing and analyzing the game, think about how the path/ramps are used and add to the intensity of the game.

Session 4: Making Clones and Creatables

When finished, students will be able to:

- Create a protagonist (user controlled), an antagonist (automated), and peripheral characters
- Base character behaviors and actions on environments and reactions to each other
- Begin to understand gaming plot and background story

Now that the students have created settings and have learned many of the more sophisticated features of Kodu, they will try to add characters that interact and build plots. There are a variety of characters in Kodu, and the interactions between them can take millions of different tracks—it is all up to the imagination of the user. Some students will be highly influenced by the gaming genres they play, so you will have kids building high impact action games while others take a more sim-like approach. It is important, however, to have a mini-lesson on how characters interact and for them to brainstorm the types of characters they want to put in their worlds. Given the depth of their interaction with Kodu, it is likely that they have already started building stories with characters, so this activity may just add a metacognitive bent on their creations.

Start by asking students about their favorite books while defining some literary terms. For instance, depending on the age of your students review the meaning of:

- Main character (protagonist)
- Opposing character (antagonist)
- Side (peripheral) characters
- Plot—the hook/inciting incident, rising action, and climax (which in literary terms consist of exposition, rising action, climax, resolution and conclusion.)

After or while you are talking about each of the above terms, project **Turtle and Fish** on the screen for students to identify the main character, the opposing character and peripheral characters. This can be particularly fun for students if one of them is in charge of the controller. Another approach to talk about the literary elements might be for student to play **Turtle and Fish** in small groups with a discussion of its elements afterwards.

See if the students might be able to identify a plot and background story. These last two literary features will be a little less tangible for the students since plot structures vary—game plots are often flat making the various plot components harder to identify.

The elements are identified as such:

Protagonist Melvin the Turtle

Antagonist Electric Flying Fish

Side Characters	Fish and Submarine

The Plot of the story:

Exposition	Ask the students about the story before the game—How is the story set up? How and why are the characters in opposition to each other? What motivates characters?
Rising Action	What moves the story forward? What events move the game forward?
Climax	What seems to be the climax in this game?
Resolution	Does the game supply a resolution or conclusion? And if not, how might it?
Conclusion	There isn't a conclusion, but students may brainstorm about how the game might end.

Reflection

During the last 20 minutes of class, have the students share their worlds again. This time perhaps have them share as a whole class. If possible, project games up on the screen and see if the students can collectively find the beginnings of the antagonist, protagonist, and peripheral characters. Through these discussions, the beginnings of plot structure may begin emerging. A big question that may be asked in this dialogue is how scoring might be used to push the story forward and add strategy to the game—currently Melvin the Turtle is just feeding the rocks back to the fish and staying clear of electric flying fish.

If they finish this activity early, tell the students to feel free to explore the other worlds that already exist in Kodu.

What You Can Expect from the Student Activity

The goal of this activity is for the students to start thinking of their games as more than simple racing and shooting environments. Games are often a complex network of interactions, and while a few stories are fairly flat in structure, most have characters that are in opposition to each other, have alliances, progress to a particular goal. Pushing the students to think of their work as a form of text may help them design more complex worlds and more complex code. Consider using the supplemental activity to further bring the students into complex story creation.

Student Sheet Activity 4: Characters and Plot

Objectives: add characters and brainstorm and perhaps begin implementing a plot structure.

Directions: In previous lesson, you created a landscape. If you haven't added characters yet to your landscape, now is the time to do that. Before you start adding or revising your characters, give some thought to the way in which the characters interact.

Complete the TO DO list and discuss with your classmates the ideas you have. (Feel free to consult Kodu to review its bot tiles for character ideas.)

To Do Check List:

- ❑ Character Brainstorming
- ❑ Protagonist (Main Character)
  - Who or what is your main character?
  - What motivates the character?
  - What are its likes and dislikes?
  - Does it have emotions?
  - How does it communicate with others?
- ❑ Antagonist (Opposing Character)
  - Who or what is your protagonist?
  - What motivates the character?
  - What are its likes and dislikes?
  - What are its emotions?
  - How does it respond to the main character and why?
- ❑ Peripheral Characters (side characters)
  - What or who are the side characters?
  - What role do they play—do they support the main or opposing character?

After you have answered the above questions, start putting the characters into the world you started creating earlier. If the setting that you made doesn't seem to fit your characters, revise it or chose a new landscape to alter or build.

Challenge Activity

Give a character a way to speak through a dialogue box—these are like the bubbles in cartoons in which characters say something. This is found under the “Say” in the programming tiles. How might use this feature effectively in a game?

Supplementary Activity

If your students are finding it hard to come up with an idea for their game worlds, consider providing them with creative prompts. Below are set of first lines (some adapted) from literature.

Seeds of Creation
Kodu, having lost his way in a gloomy forest, and being hindered by wild beasts while ascending a mountain, is met by a bot, who promises to show him three worlds…

1. Once upon a time…. It was a dark and stormy night….

Kodu was beginning to get very tired of sitting by the river with his sister, so….

This is Kodu. He is a good little creature. And always very curious…

The Koduians, after a seven years’ voyage, set sail for home, but are overtaken by a dreadful storm. The storm sinks all the ships, except one. Their leader demands the storm to stop, and the waters calm. But the winds have driven them off course to a land of friends and foes.

Kodu, obsessed with traveling through time, builds himself a time machine and, much to his surprise, travels over 800,000 years into the future. The world has been transformed with a society living in apparent harmony and bliss. But as Kodu stays in this world of the future he discovers a hidden evil….

At the last red sunset, a black line of low hills showed up in the distance. I saw a creature in the distance with its two…

The villagers of Little Leatonia still called it “the Riddle House,” even though it had been many years since the Riddle family had lived there. It stood on a hill overlooking the village, but not looking as grand as it once did. All Leatonians thought the house was creepy since half a century ago something strange and horrible happened there.

During the whole of a dull, dark, and soundless day in the autumn, when the clouds hung low in the heavens, I had been passing alone through a depressing part of the country. Suddenly, I found myself within view of the strange House of Unsure.

“Now,” said the king, “I have faith in this quest. Knights of the Blue Guard depart. I am sure I will never see all of you together again. Off now to the meadows of Galahad to save our people from doom. Rest tonight since a long journey filled with strange and wondrous creatures and places is in your future.”

Name:

Directions: Complete the following questions and then hand in your thoughts.

Setting (what does the landscape look like?)

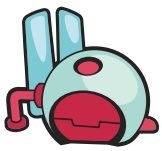
Mood (whimsical, dark/brooding, does the pace of the game go up and down?)

Characters (behaviors, conflicts, friends, alliances?)

Objects (do the trees, buildings, rocks, etc. hold a special function?)

Plot (how does the story progress?)

Why would someone want to play the game? What will make it unique?



Session 5: Starting Unique Stories and Characters

When finished, students will be:

- More attuned to strategy making
- Understand the influence of mood and tone on game play

Although the students have likely been implementing strategic designs in their games, as well as enacting strategies as they play each other's games, they should be made aware of this element of play. Strategy is the approach the player takes to win the game, and programming strategy is designed by the game author to allow for varying ways for the game to be won or lost. In any game, there are likely to be several ways for the game to be won.

Explain to them that many of the strategies the game designer included were intended to make the game challenging and interesting. There are usually many ways to play a game, and while luck often plays a role in winning or losing, strategy requires thinking through the best method to an end goal. The more a game designer can anticipate and build in the strategies and provide adjustments to make the action more or less intense, the more successful his or her game is likely to be When establishing a strategy in a game, the designer might take into account the abilities of the player's avatar, the environment, the task, and the protagonist and peripheral characters. . Talk with your students about the various ways strategy might be built within a game. Use their past projects and the other games that they have been exposed to during the course to talk about strategy.

Mood and Tone

Mood and tone can add a lot to the user experience. The media and settings in Kodu can change mood and tone drastically. If students have not explored these features, you should point them out. A change in settings can change the difficulty and intensity of a game a good deal. Under the Setting Tile, you can change the sky, lighting, water, and numerous other effects. From the various games that the students have explored, ask them how setting affected the mood. Ask them how sound and music within games influenced the feel of the game. As students talk about various games, bring the environments up to establish contrasts that will spur further dialogue. Show student how to change environmental settings, as well as sound and music. This can be done by looking at the code.

Student Sheet Activity 5: Mood and Tone

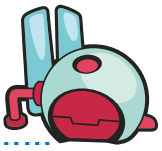
Objectives: Revise mood and tone based on strategy

Directions: In class you have been creating your own game worlds. Strategy, intensity, mood and tone can always be improved.

Given what you have talked about in class, think about how your game can be improved and the tone and mood might be changed.

Things to Consider:

- Do you think the player has a fair chance in the game—is it winnable by skill or luck? If it is more by luck then you may want to revise the game.
- What are the strategies a player can use to work through your game?
- Does the action of the game bring you into the game world?
- Are music or sound effects used in the game? Are they overly annoying or do they create tone or mood that engages the player?



Session 6: Strategy, Mood and Tone

When finished, students will be able to:

- Understand cloning and creatables

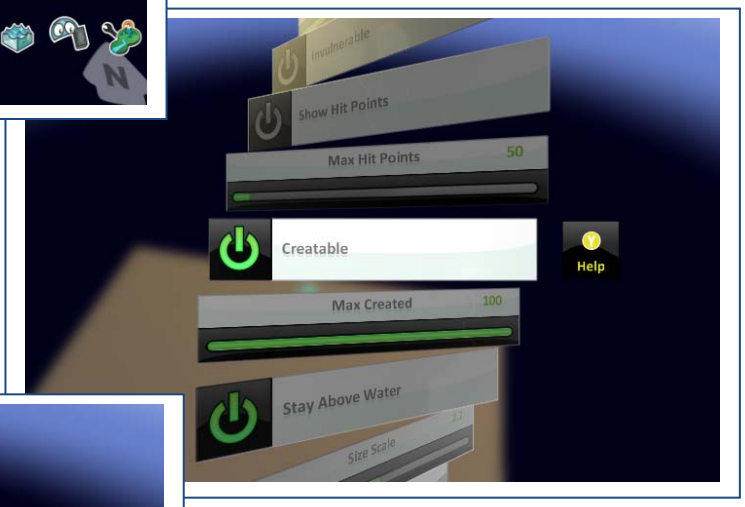
This lesson will take advantage of contrasts between copying and **Creatables**, and it will use frustration as a key window into learning. Creating armies, groups, hoards, gaggles, swarms, etc., is easy in Kodu. However, giving individuals within the groups the same set of qualities is not readily intuitive and can be time consuming when using the cloning feature alone. Usually, students start by cloning (copying characters) and then soon become tired of programming the same character attributes multiple times should they want to edit the group's behavior. **Creatables** allow the user to make the change once, rather than multiple times as one would have to do if they simply made copies of the character.

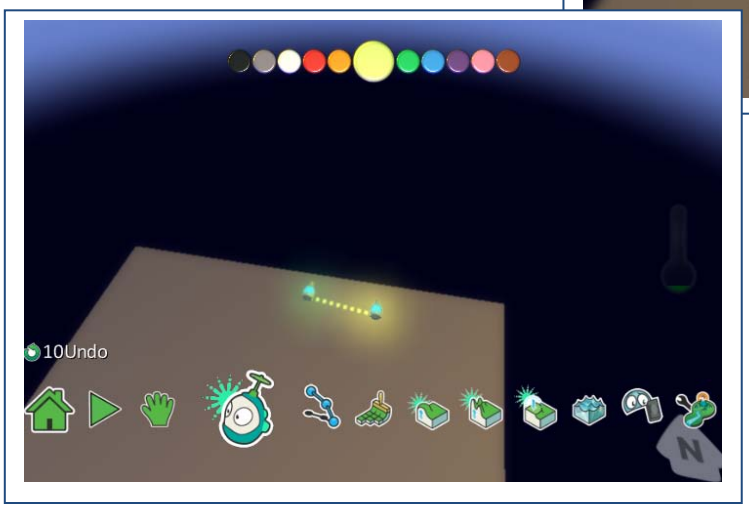
The in-class lesson will make them use the clone feature for the blimps and then try to modify the behavior of the group—this is not easily done because for the five blimps that they make, they will need to change five different lines of code. As a response to this frustration, you will then walk them through how to use **Creatable**s in which code can be changed just once. Have the students run the controls as you talk about what to do. As a whole class or in groups, have the students follow the steps below.

Cloning and Creatables

- ❑ Go to **Blimps and Jets**. You will notice that there is one blimp on one side of the field and a jet on the other. The task of the class or small groups is to make a battle in which the player controls either the blimps or the jets.
- ❑ First, program the blimp. You will notice two lines of code for the blimp. The dramatic music is coded to the blimp when it sees the jet at a distance, and the blimp glows when it gets close to the jet. Program the blimp so it moves using the keyboard.
- ❑ After you have revised the code for the blimp, make up to five clones/copies of it using the right click on the mouse and selecting **Copy**. Put the cursor in the position where you want the copy of the blimp to go and then right click again and choose **Paste** (Blimp). Make four more blimps, using the **Copy** and **Paste** method.
- ❑ Next, make sure that the blimps can move forward.
- ❑ Now give the blimps the ability to shoot blips. How many times do you need to change the code?
- ❑ Ask the class if there is a way in which to make it so you can just enter code in one location. The answer is **Creatables**.

- Demonstrate this by making a second set of blimps that operate automatically without user input. Have the student put the new set of blimps somewhere else. Make the blimp green or another color to distinguish it from the first set. Make it glow a different color as well.
- Once the new blimp is created, right click and select **Change Settings**. Find **Creatable** in the list, and turn the setting on. Settings and scroll down the list to **Creatables**. Select this option. Then, return to the blimp and copy it. If you put the cursor over one of the blimps, a perforated line will appear between the two blimps, indicating that the blimps are part of a creatable chain.





- Feel free to copy more blimps from the main blimp, which creates more creatable versions of it.
- Lastly, add code to the “main” blimp, e.g., shoot blips at jet when close by. Show the students how this code transferred out to all the blimps. When you play the actual game, the main Creatable will not be on the screen—it only appears during edit mode.
- Note to students that they should not add too many characters to their games since it can put the program in danger of crashing.

Subject Area Link—
Social Studies & Science

The use of Creatables is powerful, and it can create an impressive effect of a legion of troops all moving the same fashion. Consider how the tool might be used to design the actions of a famous battle while using primary and secondary sources as the research base.

Interestingly, Kodu can also be used to create a more naturalistic look. For instance, think about schools of fish with each school being its own Creatable set programmed to wonder. Then within each set, you can program a set of attributes unique to the school of fish.

Student Sheet Activity 6: Creatables

Objectives: Make clones and understand the premise of Creatables.

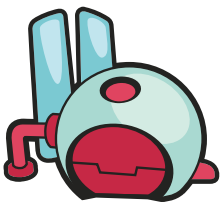
Directions: Follow the To Do below and have your instructor check out your progress before moving onto the extra activity. This is a fairly open-ended activity so feel free to be creative in your use of Creatables.

To Do Check List:

- ❑ In a world you have already crafted or in a new game all together, add two or three sets of characters, like we did with the two sets of blimps and the set of jets. (Creatables do not necessarily need to march in unison. Use **Move**, **Wonder** to code a more naturalistic affect.)

Extra Activity

If you finish and you have time to spare, check out Technique: Launching Creatables v02. Note how the apple has to be made into a creatable in order for Kodu to launch it. Take the apple away, and Kodu can launch nothing. Can you reason through why this would be?



Session 7: Changing Behaviors Using Pages, Establishing, and Shifting Perspectives

When finished, students will be able to:

- Understand the idea of pages, when they are used and why
- Use nearby and far away features
- Shift camera angles in settings and in the code

This lesson is based on discovery. Have students play **Vendura v14**. After they have played the game for awhile, ask them to look at the world and see if they can determine how many objects and characters comprise the world and what each object or character does.

Here's a list:

Turtle—user controls, acquires wisp shooting ability by interaction with Kodu

Stick—informs the turtle (the user) that Koduwan has been waiting

Kodu—instills power and disappears

Wisps—given to turtle as a power

Castles—unleash mines that shoot

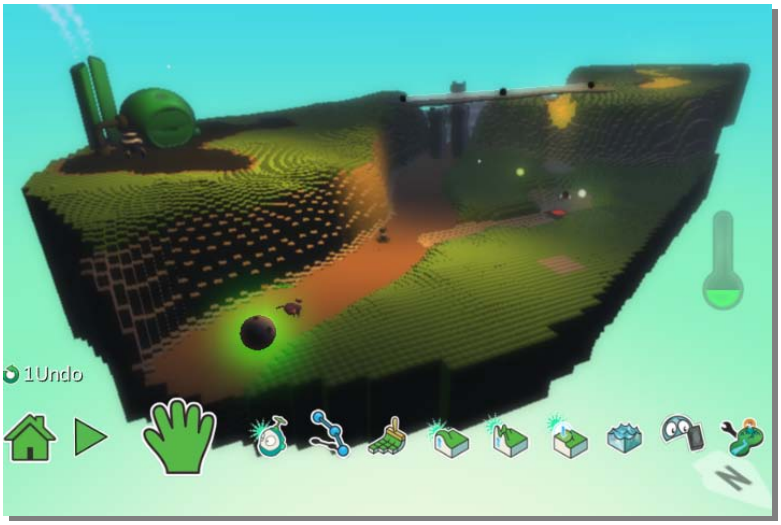
Hearts—creatable that act as food

Mines—move randomly and shoot at Kodu

Factory—acts as the object that triggers the end

Pushpad—informs the turtle what to do

If the students have not pointed to the code, assign them an object or character to evaluate. They should be tasked to report back to the group how the object or character is coded. Using the projector, have them present each of their findings. During their presentation, pay particular attention to pages, and secondarily to how nearby, far away, and camera shifting are used.



To help guide the discussion, consider how in **Vendura**, objects and characters provide the player with an understanding of what's going on. They act as directions for what to do next. However to create the effect of shifting action, pages are necessary. Pages help shift behaviors either based on time or in reaction to some occurrence.

Pages

The stick directing the turtle to Koduwan is the first use of pages. The code changes the state of stick from closed (page 1), which is its constant state, to open (page 2). Page 2 then expresses another set of behaviors which run and then return to the initial state of closed (page 1) after a certain amount of time. The second use of pages comes with the turtle's approach of Koduwan. Page 1 supplies a resting state for the Kodu. Page 2 activates a behavior, specifically Koduwan turning and greeting the turtle. After three seconds, Page 3 is enlisted. It supplies another set of text to carry the story along. After eight seconds, a wisp is created and given to the turtle as Koduwan disappears in a boom.

This is a demonstration of how Pages work—they allow for a change in character behavior. Often the change is triggered by an event within game play. For instance, eating an apple may allow a character who is typically chased by another character to suddenly be the aggressor. This change in behavior can be temporary or permanent depending on how the program is coded. Pages can be added by entering Program, and then toggling to a different page by either clicking L or R at the top of the Kodu window.

Close-By and Far Away

How dramatic effect is created through this series of exchanges and behaviors should be pointed out to the students. While pages help facilitate the operations, the close-by and far away tiles allow users to create an element of surprise. In the entry sequence, both the stick and Koduwan's behaviors are guided by sensing the turtle nearby and then shut off based on the timer. This is effective for moving the story along. This type of effective use should be mentioned to your students. Camera Shift

Types of Player Perspective

First Person – Players view the game through the eyes of the character

Third Person – Players exist within the game world which is less immersive

Top-Down – Player has a global view of the game world

Side-scrolling – Fast paced; action is viewed from side-view camera; doesn't show a lot about the world

Isometric – Player has a global view

Also, point out how the camera perspective shifts when the turtle hovers on the red land square. When playing, this shift in perspective seems to be caused by the turtle's approach to Koduwan. This is an effective use of land since the camera jars the players into a different perspective. Camera positioning is an effective tool for the creation of a particular effect. Perhaps have the students think about the difference in effect between first and third person—when and why would one use one or the other? While the camera can be changed using the tiles, it can also be changed under settings in the main menu.

Extra Activity

Have students reinstate the original perspective.

On Page 2 write:

When: see, Kodu, none, close by Do: Switch Page 3

On Page 3 write:

When: see, Kodu, none, nearby Do: open, once

When: see Kodu none Do: switch, page 1

Introduction to the Game Project

Close the session by talking about the final project for Kodu. The criteria for the project are up to you, but by introducing the project now, the students will have plenty of time to generate ideas and to work on their projects. Be sure to have students plan by drawing out or planning their games through a brainstorming web, narrative free write, or some other format to get the creative juices flowing.

Student Sheet Activity 7: Camera Angles and Shifting Behavior

Objectives: shift camera perspective, use close-by and far away, implement shift in behavior using pages.

Directions: You have just looked at code and techniques for creating dramatic effects and for crafting changes in behavior within a character. Now it's time to practice.

To Do Check List:

- ❑ Consider the three things you talked about during today's lesson—close-by and far away, camera angles, and pages.
- ❑ Explore the various games you have in your Kodu deck and how you might implement the three operations within a single game. For instance, look at **Chaotic Orbitals v3** and code in a camera shift to first person, maybe by holding down the **Spacebar**. The camera shift might also be initiated by bumping, come close to, eating, or grabbing the coin.
- ❑ Now, add characters that change behaviors based on their interactions with each other or objects in the world. To create this system of interactions, you may need time to brainstorm and play with the tiles to see what is available to you. If you need assistance, consider adding character and objects in **Chaotic Orbitals v3** that have a resting state (Page 1) and then are spurred on to another behavior based on interactions with other characters.

Extra Activity

If you finish the above activity and want another program challenge, go to Technique, Change Behavior. First play the game and see if you can 'win' the game. Next, program the game so that you view game play from the perspective of the cycle.

OR

Work on your own game world.

Session 8: Power Ups, Health, and Timer

When finished, students will be able to:

- Use timers, health monitors, and power ups

In the games that we have used in lessons and that you may have had students explore on their own, you may have seen timers, health monitors and power ups being used. In fact, some students may already being using these features in their play and design of games. This session will look at these three areas a little more closely.

To start, discuss with the class the concept of power ups. In all likelihood they will have a fairly good grasp of how they operate. Basically, a character eats, grabs, bumps, etc. something that gives it a special skill or ability for game advantage—often this is temporary. There are also power downs which have the opposite effect. **Technique: Change Behavior** is an example of a very simple power up—by eating the apple, the bot is able to jump. This might be demonstrated on the projector. Some students may have experimented with this game already.

Now, have the students explore two games that utilize power ups, at the same time they are also connecting to scoring and health. Tell the students you want them to play each of the game and identify the power up (or down) and also identify the code associated with each. After they have played the games, have the students take turns demonstrating and talking about how the power up operates in the game.

It is also important for the kids to start thinking critically about the games. For instance, encourage them to play a Siskel and Ebert type dialogue to help facilitate the discussion. It would be fruitful for them to talk about how the power ups/downs might be better implemented and lend themselves to strategic game play. If they don't enter into this discussion readily, consider asking how types of land, timers, allies, thieves, etc might play a role. No special powers are associated with the acquisition of the coins, apples and hearts in these games, which begs the question of how power ups and downs might be better implemented.

Health monitors do play a role in the games. Ask them how these add a second strategic element beyond simple scoring.

The two games are:

Rock Fight KB—the player needs to grab hearts for ammo and loses points for shooting; the health monitor goes up has hearts are grabbed and down as ammo is released. Use arrows to move and left shift and z to launch rocks

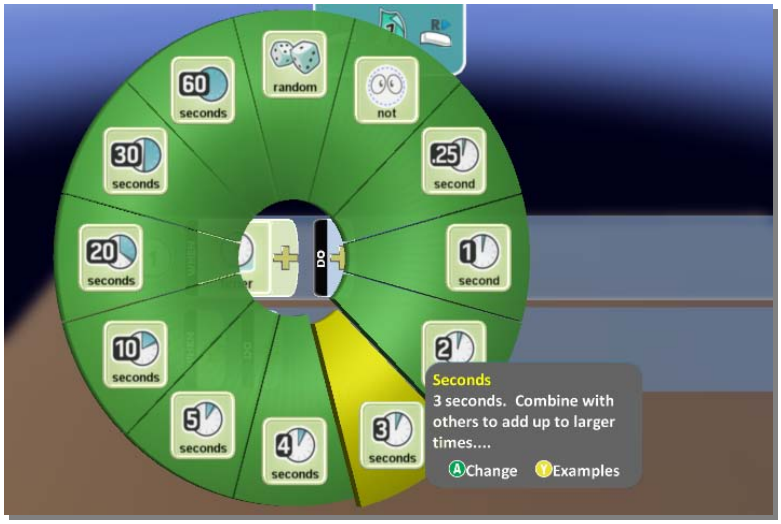
Pandemica KB—the player needs to eat apples to accumulate points and be healed

Establishing Code for the Timer, Health Monitor, and Points

These three tools can boost communication within the game by signaling the condition of the characters, as well as show how they are performing in the game world. Obviously, points are crucial to winning or losing a game, but they can also be used to facilitate the giving or taking away of powers or abilities. All these tools can be found on the main tile wheel under **Program**.

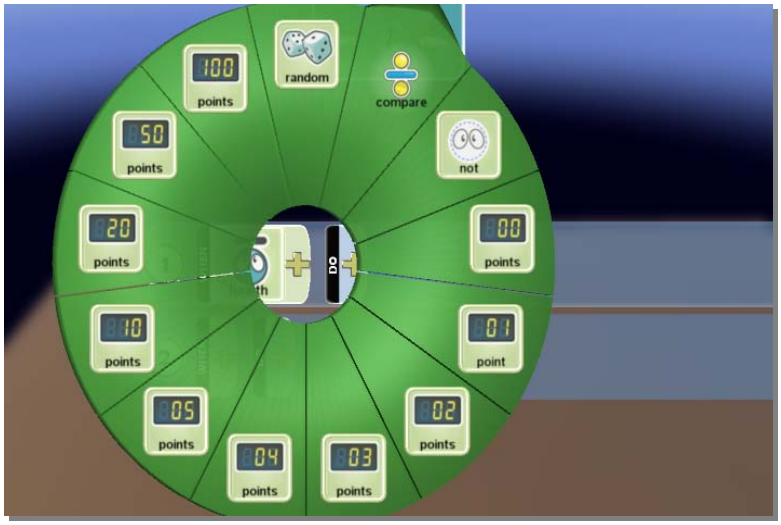
Timer

The **Timer** can be used in a myriad of ways—beating the clock game scenarios to establish how long a character/player either does or does not have ammo, a power or a tool. The latter can be facilitated by working between Pages (previous lesson), in which the code of behaviors for a character/player is bound within a time limit.



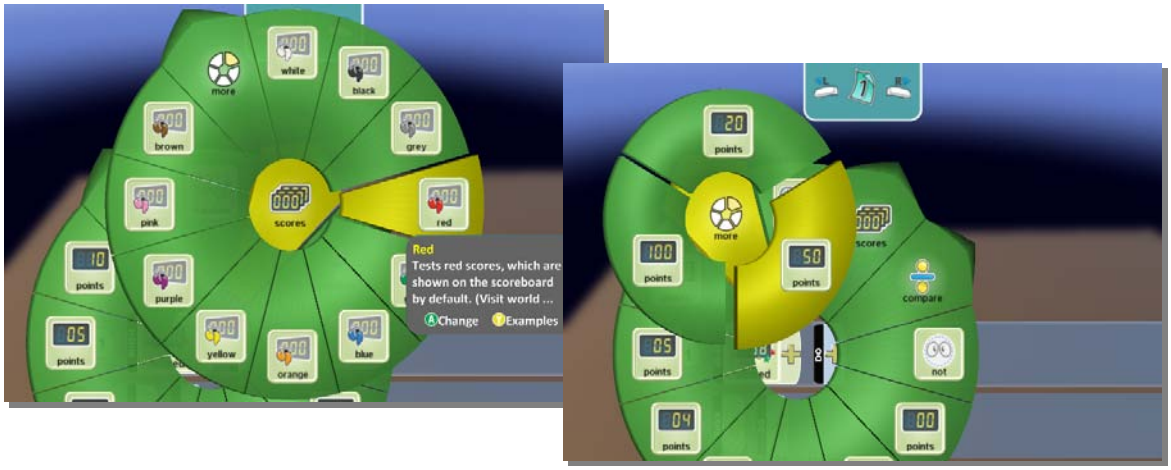
Health Monitor

The **Health Monitor** is an effective way to communicate information to the player about his or her character, as well as the other characters in the field of play. They can also facilitate changes in behavior. For instance, when the health of a character falls below or above a point limit the code might switch to another set of behaviors that push the game narrative or play forward. Health can also be set in comparison to a set point total or another character's health points.



General Points

The point systems of a game world can be as complex or simple as the user wants to make them. Character point tallies are distinguished by designating color in the code.



Game Creation

Have the students continue to develop their game worlds. As they begin to develop them, encourage to think about how health, timers, and point systems might work in their world. The students do not have to include all these elements, especially if their game is more narrative based, but they should begin to think about how one of the elements might be incorporated into their game worlds.

Reflection

During the last 20 minutes of your session, have the students share what they programmed with one another. If possible, try to verbally note the different ways in which students tackled the problem. Or try to surface the issues that students were having as they created their power ups. The intention is for the sharing of problem solving. Have them journal about either their own game or that of another.

Subject Area Link:

Math experts know that everything is math—or at least can be turned into math. Kodu offers a number of ways in which an instructor can stress applied mathematics while students craft behaviors and engage in design.

\While coding naturally lends itself to the concept of branching, the features associated with health, timing, and points can put mathematical reasoning into overdrive, especially they are used to facilitate an action or behavior and communicate information.

Built into the tools are systems of comparison, which coordinate greater and less than problem solving.

- Do you think the player has a fair chance in the game—is it winnable by skill or luck? If it is more by luck then you may want to revise the game.
- What are the strategies a player can use to work through your game?
- Does the action of the game bring you into the game world?
- Are music or sound effects used in the game? Are they overly annoying or do they create tone or mood that engages the player?

Student Sheet Activity 8: Timers, Health and Power Ups

Objectives: Use timers, health monitors, and power ups

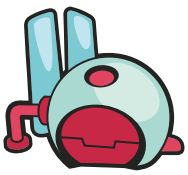
Directions: You have just looked at how power ups operate. You have also identified how health monitors are applied to characters to activate another level of strategy within game play. Now, it is your turn to implement power ups.

To Do Check List:

- ❑ Use either your own game world or an existing world to create a system of power ups and downs.
- ❑ As you are considering how to initiate your system, think about how land, objects, and other characters might be involved.
- ❑ To add a layer of challenge, try to make the power up or down temporary—meaning put it on a timer.

Extra Activity

As you and your peers develop games, play and offer feedback to each other. Your instructor will be asking you to reflect on either your or another's game and how strategy is developed within it.



Session 9: More on Scoring - Basics to Communication

When finished, students will be able to:

- Use scoring in more complex ways

Your students have likely been using scoring while playing and developing their own games. If they haven't been paying much attention to it as a tool to control behaviors, then this lesson will be beneficial for you and your students.

Create a new level for this lesson. You can call this Scoring Lesson. Use a simple landscape. Add a Kodu and at least 4 cycles. Program Kodu to shoot on command. Kodu should score 5 points every time he shoots a cycle. You can create this code in front of students. Ask them how many cycles you will need to add before Kodu can win the game (answer is 4). Now add an enemy for Kodu that can damage it in some way. Add wisps that move randomly and subtract points from Kodu's score. Ask a volunteer to add these wisps. They will have to change the program for Kodu so that his score changes when he is hit by the wisp. Ask them to change the program for the wisps so that they are attracted to Kodu once he reaches 15 points. This will make the game more challenging.

Have students play **Bonk Out v.18** with their groups. After playing this game discuss if the game was fun. Have them look at the code and discuss how the timer and points are used in the game. Ask how the action of the game compares with other games that also use points, health meters and timers. The key to understanding scoring is to notice that is not just a method to track who wins and who loses, but it can also be a method to adjust behaviors and to communicate.

Game Creation

Students will spend the remainder of the session working on their games for their final projects. During the last half hour of class, have students share their work offering feedback to one another.

Journal reflection

End the class with a short journaling exercise. Did you create your game with an audience in mind? What types of people do you think would like your game most? Is your system of points or a timer involved in the construction of the game? How are they used? Based on watching others play your game and the feedback you have gotten, is your game too challenging or easy? What makes you think so?

Student Sheet Activity 9: Scoring and Behavior

Objectives: Use scoring to change a behavior

Directions: You will now see how to add scoring to your game – you can use different colored scores to track different things. Then, have an action in your game be taken based on the value of a score.

To Do

- ❑ Choose any game environment you want to compete the tasks below
- ❑ Create a simple game in which points are given for doing certain things—like eating, bumping, holding/dropping, etc.
- ❑ Design a situation in which a competitor either subtracts points from your score or has its own score tally comprised of a different color
- ❑ Code a system in which an action is taken or not taken based on a set of scores

Extra Activity

Work on your game world for your final project.

Kodu Finale: Student Presentations

Student Presentations

This entire session will be devoted to student presentations. Outsiders (parents, game developers in the community, other students) will be invited to come play the games that students have created. This is a great opportunity for students to showcase and share the games that they worked to create.