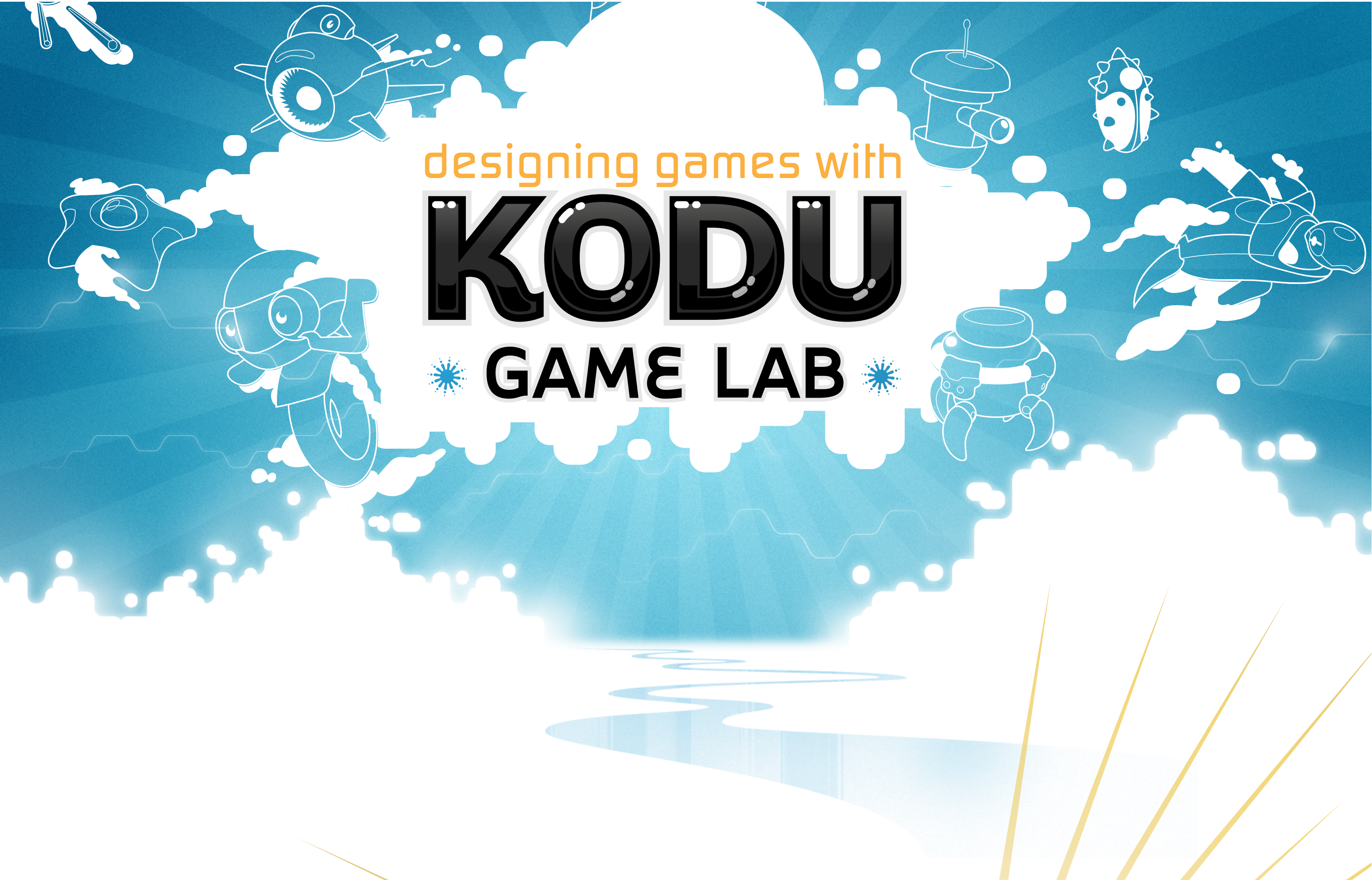


designing games with

KODU GAME LAB

WORKSHOP TRAINER MANUAL

Microsoft

PRESENTED BY

ideaslab

EXPLORING NEW POSSIBILITIES

Contents of Workshop Manual

Introduction

Kodu Game Lab is a visual programming language that allows anyone, even those without programming knowledge and experience, to create rich 3D games.
Kodu uses intuitive icons to control the characters and objects with character behaviours expressed in physical terms.

The Creating Games with Kodu Game Lab course is a designed primarily for teachers looking to introduce game design into their classroom program. This two day intensive course will introduce Kodu Game Lab and demonstrate how anyone can create rich and exciting games.

The course will cover the skills needed to design and create worlds and games with Kodu Game Lab and will also investigate the design process and how Kodu can used in the school curriculum.

Through designing and creating their own games, participants will learn how powerful and interesting games can be created with simple building blocks and techniques.

The course will:

- enable participants to program characters and objects
- enable participants to build 3D worlds
- introduce the key elements of games design
- provide a rich range of code examples
- provide a rich range of game play examples
- provide examples of how Kodu Game Lab has been used in schools
- give plenty of hands on time so that participants are confident in using Kodu Game Lab

Important Note

Kodu Game Lab is still under development. Every effort has been made to make sure that the examples contained in this manual work in the current version 1.48 of Kodu Game Lab. When using other versions of Kodu Game Lab please check http://media.planetkodu.com/workshop/resources.html for updates and workshop materials.

This is the trainer's manual. Trainer only notes appear in the boxes throughout. These are not included in the participant's manual.

Module 1
Introducing Game Design and Kodu Game Lab

Activity 1.1 Introduce yourself

Use this activity to enable the group to get to know each other. Allow ten minutes for the group to create their characters and then ten minutes for the group to share with the rest of the groups.

If you were a game character what would you be like?

Name: ____

Powers: ____

3 things that make your character special:

____

____

____

____

Draw your character in the space below:

Activity 1.2 Introducing Kodu Game Lab

Key points to communicate:

Kodu is for rapid game development using an easy to use interface and language

Kodu uses an icon based programming language or physical metaphors

Kodu is for small games

During this activity we will explore Kodu Game Lab, its purpose and what others have created with it. Use the space below to record any important ideas about Kodu Game Lab.

Activity 1.3 What is a game?

Ask the group to think of five games that they like to play. Encourage them to list different types of games, for example, a sport, a board game, a computer game. Once they have chosen five games ask them to think about what these games have in common, and what makes a game. Use the Diamond 9 to display the key factors about what makes a game with the most important factors at the top.

Participants should share their ideas at the end of the activity.

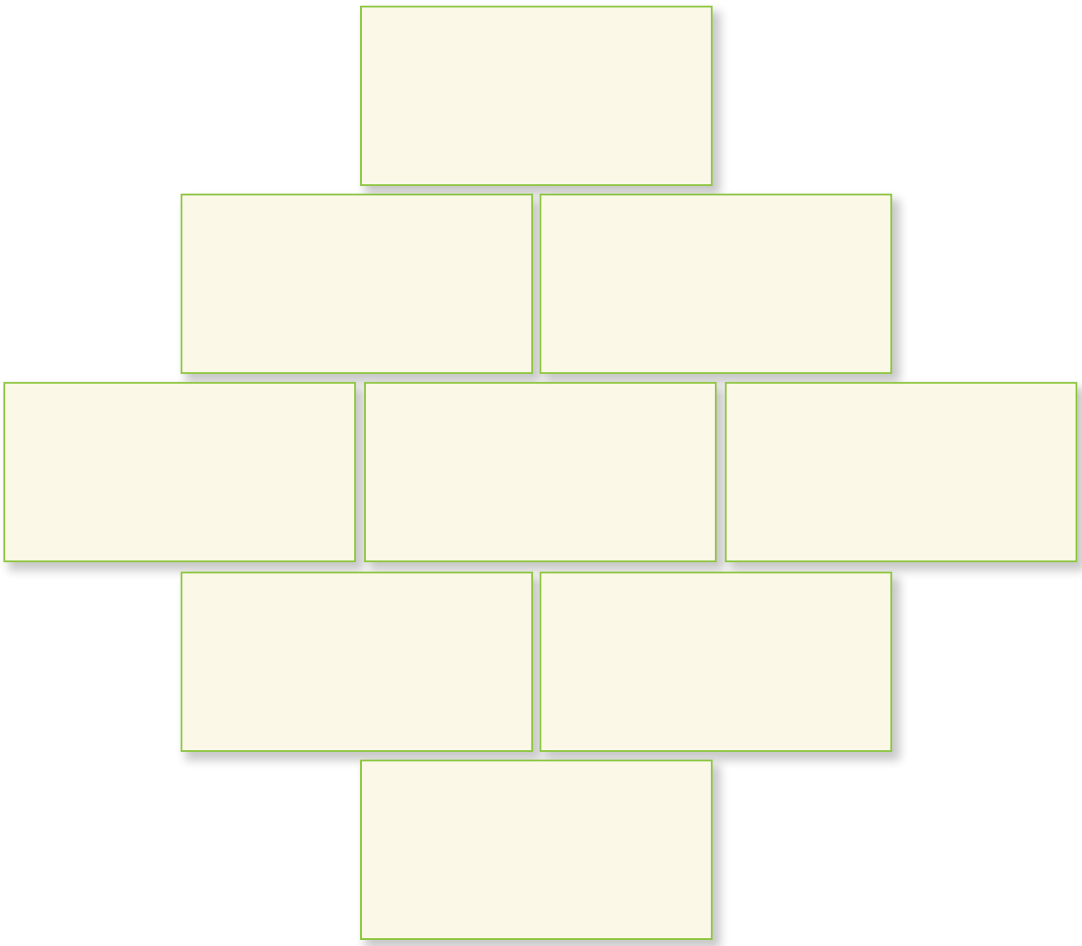
Key points to highlight:

Games are fun.
Games have rules.
Games have an objective / Games have winners and losers.
Games require players to make decisions and require skill.
Games are an experience.

In this activity we will explore what makes a game, a game.

List five games that you like to play:

Thinking about the games that you have listed, identify the key factors that make a game a game. Use the Diamond 9 graphic organiser below to show nine key factors of games with the most important factors at the top of the diamond.



"If you've written a software subroutine that takes more than ten arguments, look again. You probably missed a few. "
Alan Kay

"A game is a problem-solving activity, approached with a playful attitude."
Jesse Schell

Activity 1.4 Kodu Games

This activity is designed to allow participants to discover the types of games that can be created with Kodu Games Lab and identify what makes these games fun. Allocate 15 minutes for game play, 5 minutes to record their findings and then 10 minutes for discussion. Use the "Kodu Game" PowerPoint to share some of the games that have been created by the Xbox Kodu community.

As an example have a quick play of Bonk Out 18 and discuss its objectives, rules and why it is fun.

Kodu Game Lab has around fifty worlds and games included by default. In this activity we will explore the some of these worlds and games to learn about what can be made with Kodu Game Lab and what makes a Kodu Game Lab game.

Game Review (Example)

Name: Bonk-Out v18

Objective: Knock over the castles while protecting your sticks.

Rules: A button launches a pucks. Push bots can also launch pucks. Pucks destroy everything except the player. The player can control where the pucks go by bouncing them away.

Why is it fun? This game is easy to understand as it is similar to the classic game Breakout. It is also fun because it is fast moving. This game would be much better if it was longer.

Your Game Review

Game Name: ____

Objective: ____

____

____

____

Rules: ____

____

____

Why is it fun?____

____

____

Module 2
Programming Basics

Activity 2.1 Introducing Kodu's programming language

In this activity we will write our first Kodu Game Lab program. Kodu Game Lab contains three default tutorials and we will work through all three.

Demonstrate Tutorial 1 and how to solve it.

Key Points:

Physical metaphors and when do statements that define the programming language.

Highlight the in-game help menus which display which keys and buttons can be used for navigation and to access the various screens and toolbars.

Notes

Complete Tutorial 1

1. Open the **Tutorial 1** world.

The Kodu bot tells you that "**I want to visit the castle**"

2. Press **Esc** (keyboard) or **start** (controller) to edit the world.

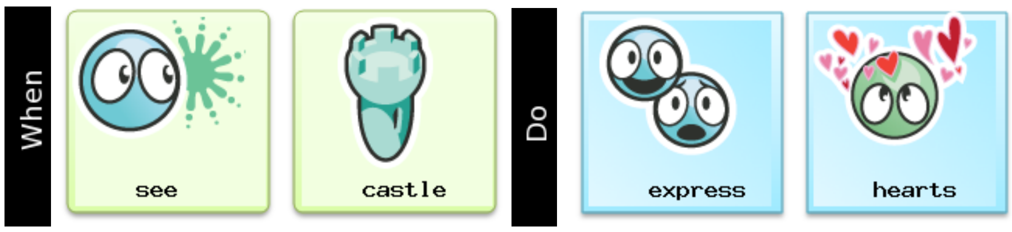
3. Select the "**Add and Program Objects**" icon from the toolbar.

4. Select the character by **right-clicking** it (keyboard) or selecting Y (gamepad). It should be now glowing yellow and the editing menu should appear.

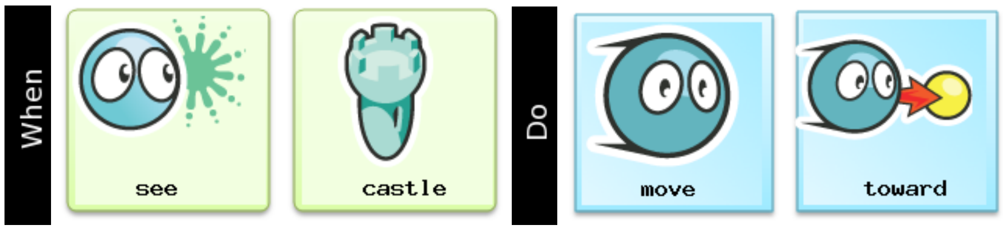
5. Select **Program** by left clicking it.

The bots code should now be displayed, we have two choices now either to edit line 1 in the existing code or add a new line to the code. We will edit the first line but adding a new line of code is also a valid approach.

The first line reads



We need to change it to



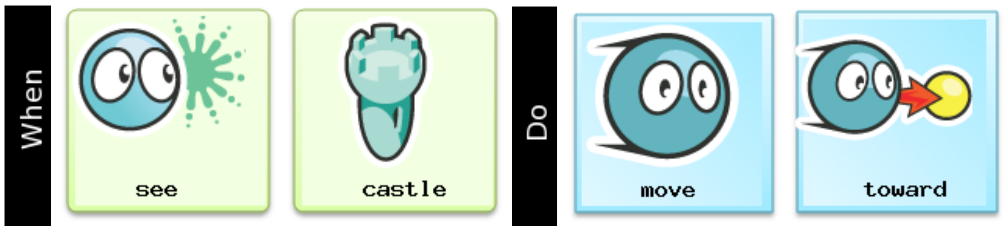
6. Remove the express and hearts tiles by right clicking on them or pressing X on the gamepad

7. Add the new tiles by left clicking (or pressing A on the gamepad) the icon to the right of the do tile.

8. Select move.

9. Select towards.

The code should now be:



9. Press esc on keyboard (or B twice on the gamepad) to exit the programming screens and return to the toolbar.

10. Click on Play to run the program or select with the gamepad and then A to play the game.

11. The Kodu bot should now move towards the castle.

Activity 2.2 Solve Tutorials 2 and 3

In this activity we will solve Tutorial 2 and Tutorial 3. The instructions to solving these programming challenges are included below but please attempt to solve them without the instructions if possible.

Ask the group to solve Tutorials 2 and 3. Encourage them not to use the instructions if possible.

Notes

Tutorial 2 Solution

Objective: We need to reprogram the cycle bot so it can jump the river.

1. Open Tutorial 2.

2. Press **Esc** (keyboard) or **start** (controller) to edit the world.

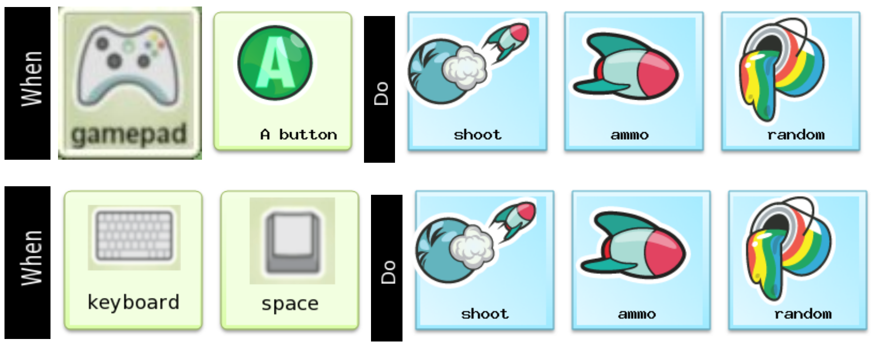
3. Select the "**Add and Program Objects**" icon from the toolbar.

4. Select the character by **right-clicking** it (keyboard) or moving the cursor with gamepad. It should be now glowing yellow and the editing menu should appear.

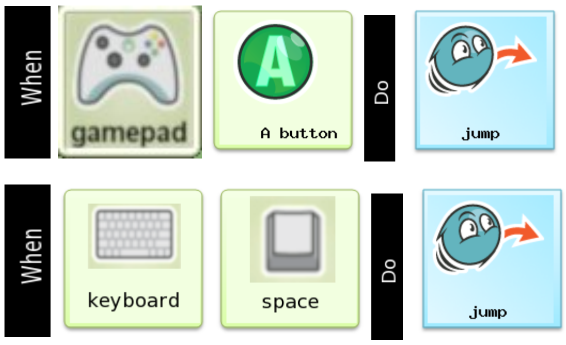
5. Select **Program** by left clicking it or with the button on the gamepad.

The bots code should now be displayed, we have two choices now either to edit line 3 and 4 in the existing code or add a new lines to the code. We will edit the lines but adding a new line of code is also a valid approach.

Currently the lines are:



We need to change them to:

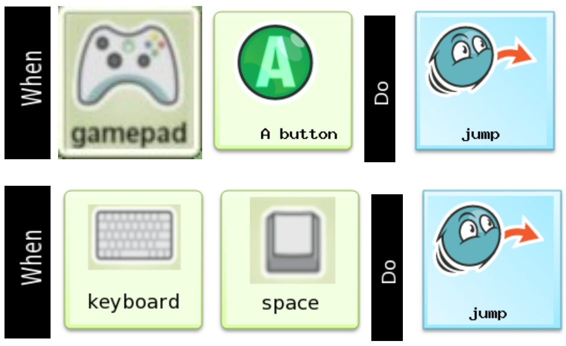


6. Add the new tiles by left clicking (or pressing on the gamepad) the icon to the right of the do tile.

7. Select using the left mouse button or by pressing on the gamepad.

Note click on actions to find

The code should now be:



8. Press esc on keyboard (or twice on the gamepad) to exit the programming screens and return to the toolbar.

9. Click on Play to run the program or select with the gamepad and then to play the game.

10. The Kodu bot should now jump with either the spacebar or the button.

Tutorial 3 Solution

Objective: To program the cycle bot to pick up an apple.

1. Open Tutorial 3.

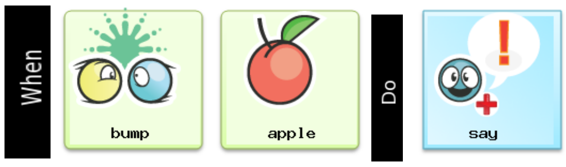
2. Press **Esc** (keyboard) or **start** (controller) to edit the world.

3. Select the “**Add and Program Objects**” icon from the toolbar.

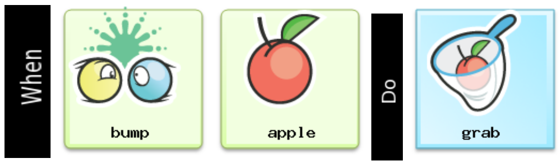
4. Select the character by **right-clicking** it (keyboard) or moving the cursor with gamepad. It should be now glowing yellow and the editing menu should appear.

5. Select **Program** by left clicking it or with the button on the gamepad.

The second line of code reads



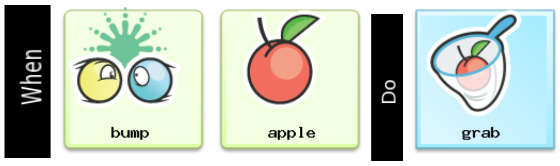
We need to change it to:



6. Add the new tiles by left clicking (or pressing on the gamepad) the + icon to the right of the do tile.

7. Select using the left mouse button or by pressing on the gamepad. Note click on holding group to find

The code should now be:



8. Press **esc** on the keyboard (or twice on the gamepad) to exit the programming screens and return to the toolbar.

9. Click on Play to run the program or select with the gamepad and then to play the game.

10. The Kodu bot should now pick up the glowing apple and be able to cross the bridge.

Activity 2.3 Getting to Know the Bots and Objects

In this activity we will explore the various bots and objects that are available in Kodu Game Lab. We will investigate the various attributes and abilities of the different bots.

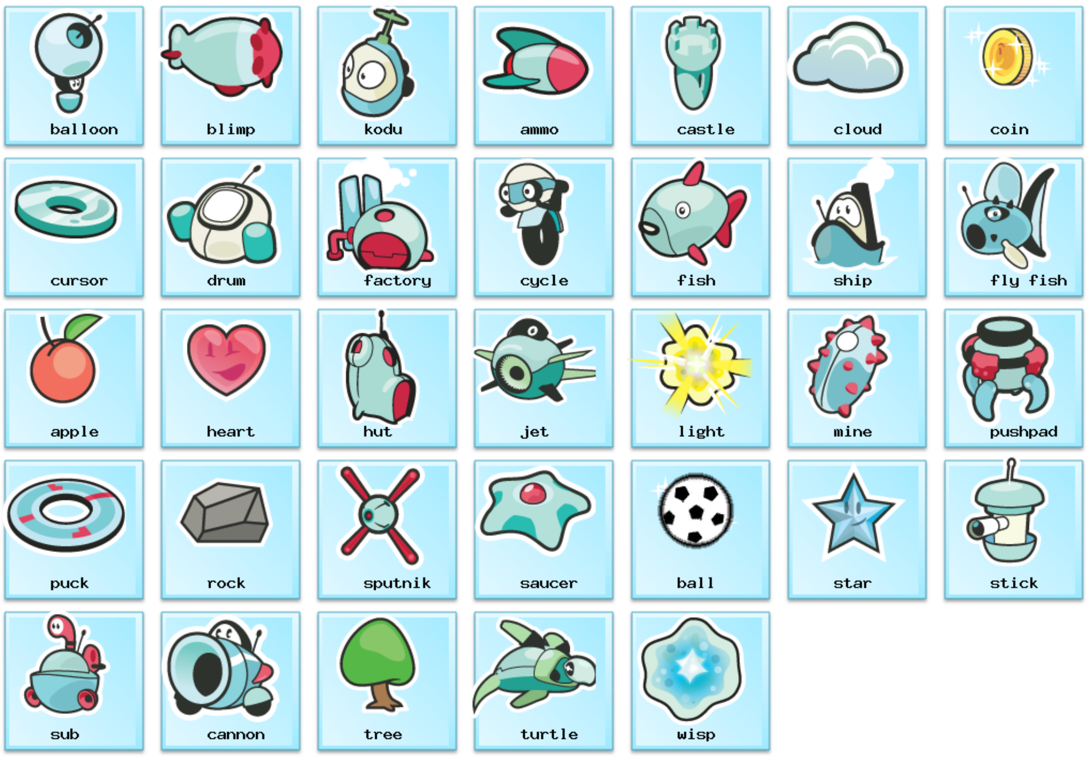
Use Bots and Objects World to demonstrate the range of bots and objects available in Kodu Game Lab.

Main Points to highlight:

1. Different bots have different attributes and abilities.

2. Use colours to distinguish different character roles eg red cycle, black cycle

3. All bots and objects have settings that can set as the game is created but not by using programming during the game.

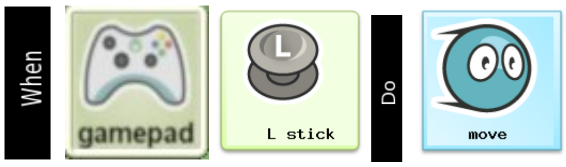


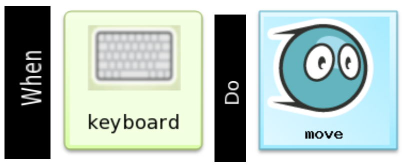
Notes

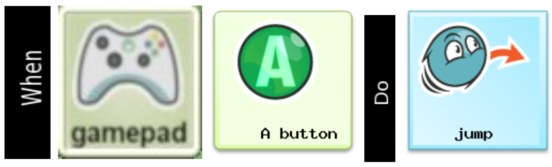
Activity 2.4 Programming Bots

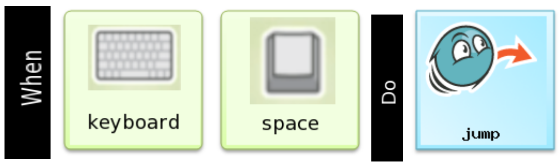
Player controlled bots

We have seen in the above tutorials that we can write simple programs in order to control a bot with the keyboard or xbox controller. We can also map actions such as jumping to specific keys on the keyboard or gamepad



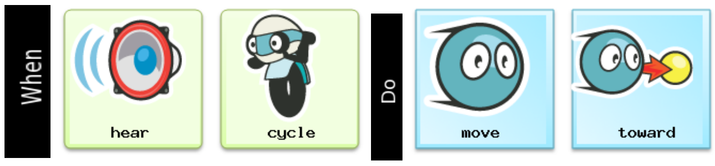




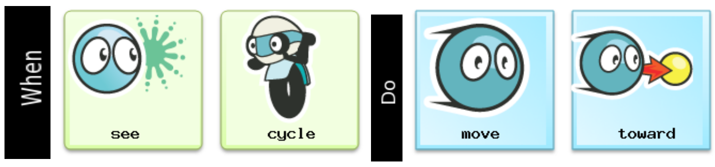


AI bots

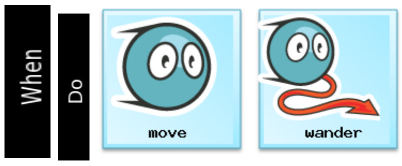
Lets now look at how we can program other bots that will interact with the player's bot.



Alternatively, we could use see rather than hear:



If we just want the bots to wander rather than follow we can use:



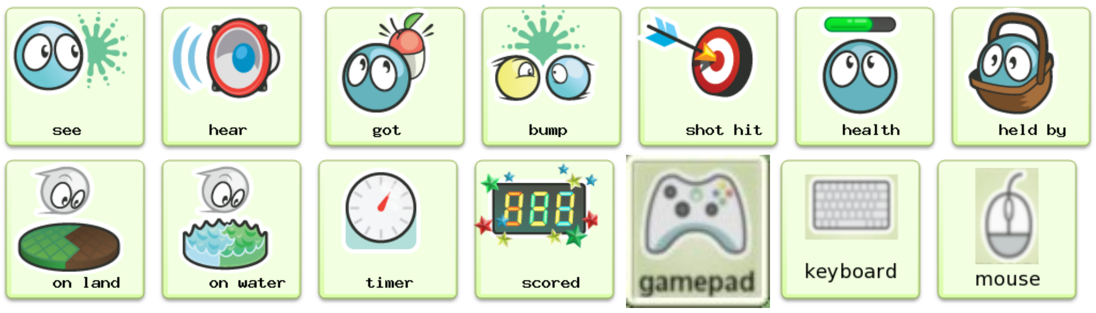
When physical sensors:

Key Points:

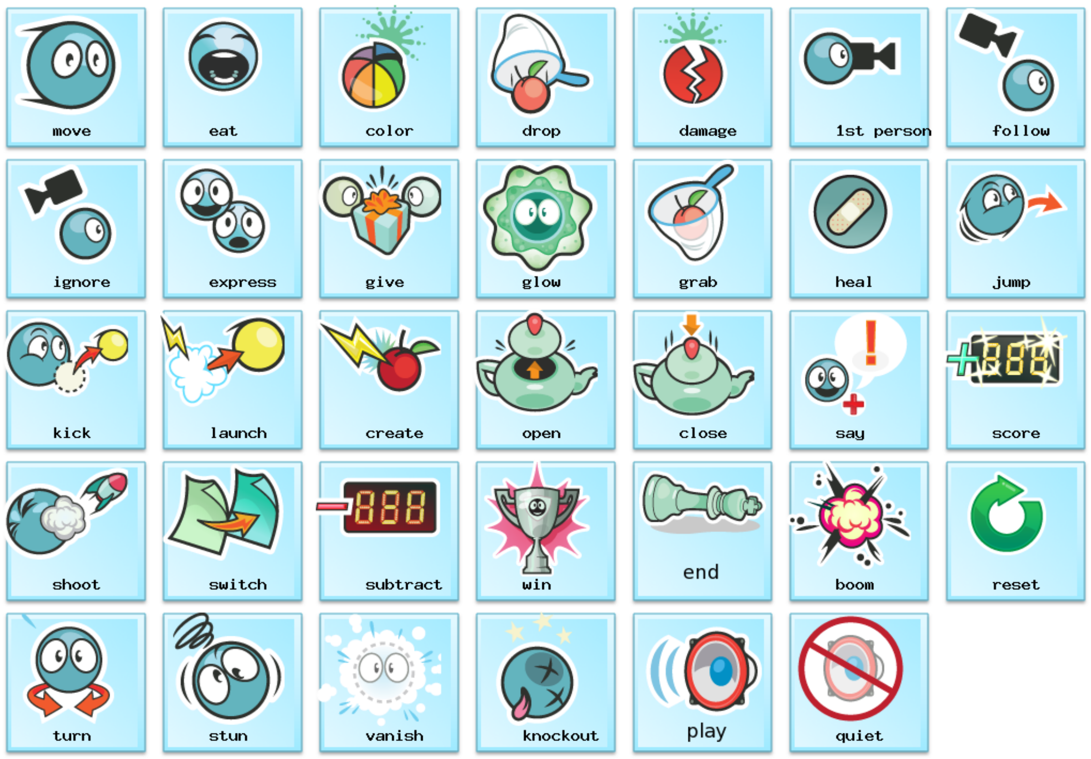
1. See, hear, got, bump, shot hit, health, held by, on land and on water are physical actions of the bot or object.

2. Timer, scored, gamepad, keyboard and mouse are external actions that the bot or object can respond to.

3. All of the actions are not available to all of the bots and objects.



Do physical actions:



Bots and Objects have settings that define certain attributes which determine how the appear and behave. Programming cannot modify these settings while the game is running and must be done in the design stage.

Notes

Activity 2.4 Create a two bots

In this activity we will make a simple two bot game, like tag or similar.

This activity is to make a simple two bot game. The game shouldn't be complicated but should be achievable in a short amount of time.

Create your first bot.

1. Open **Small World with Water**.

2. Press **Esc** (keyboard) or **start** (controller) to edit the world.

3. Select the "**Add and Program Objects**" icon from the toolbar.

4. Add the character by **left-clicking** it (keyboard) or with the Ⓐ button on the gamepad. Select the bot or object you want.

5. To program the bot or object select **Program** by left clicking it or with the Ⓨ button on the gamepad.

6. Add your desired code.

7. Press **esc** on the keyboard (or Ⓑ twice on the gamepad) to exit the programming screens and return to the toolbar.

8. Click on Play ▶ to run the program or select ▶ with the gamepad and then Ⓐ to play the game.

9. Repeat the process to add your second bot.

Notes

___

___

___

___

___

Module 3
Informal Design Process

Activity 3.1 Generating Ideas

Ask the group to think about the game they just made or tried to make and think about how it could be made more interesting.

Ask them to record these ideas.

List ideas about how to improve your game

Activity 3.2 – Game Ideas

In this activity we will generate some ideas for games we could make with Kodu Game Lab.

List a game idea that you may have:

Example

Name: Duck-Out

Objective: This game is similar to bonk out except instead of castles and stick bots boats and turtles will be used and the game will be played on water. The turtles will also be able to duck under the water for two seconds in order to avoid the pucks.

Rules: The player controls the fish. The fish must protect the three turtles from the flying pucks by bouncing them away. Player can also cause the turtles to duck under the water for a short period to avoid the flying pucks.

Bots used: Boats, Turtles, fish and pucks

Name:

Objective:

Rules:

Bots used:

Activity 3.3 School Examples / Case Studies

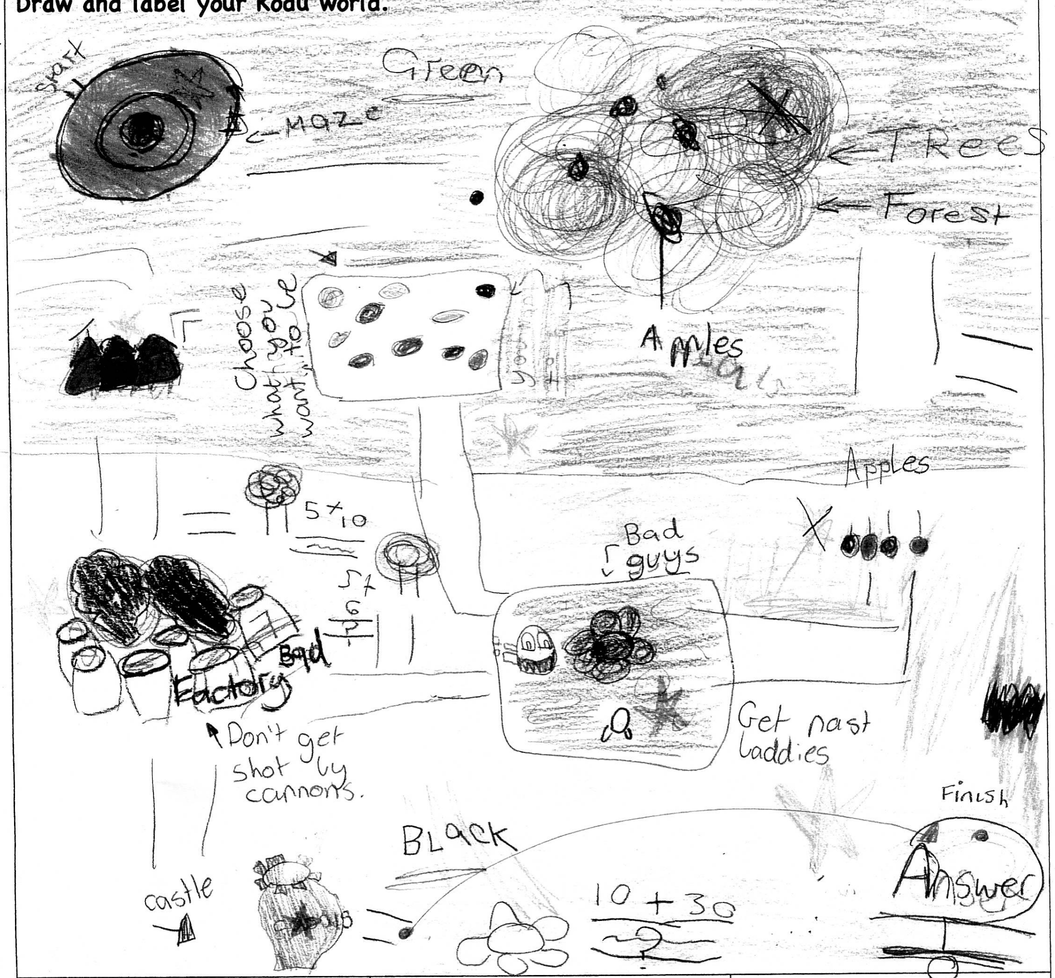
In this activity we will take a step back from thinking about games and look at how various schools have used Kodu Game Lab in their learning program.

Notes

(Rand D) Reasea and Developme 36+36= 156+156= 44+44= Time 20 Min Scor Bord

KODU GAME DESIGN TEMPLATE

Draw and label your Kodu world.



Describe the Game (and what do the players have to do?)	List the Kodu objects and their purpose.	Describe how a player wins/finishes the game, how a player scores and how long your game goes for.
1 Collect Red apples. 2 Answer the questions 3. Find Bonus stars 4. Don't get pink apples 5. get past bad guys 6. don't get shot by cannons. 7. Get star in castle 8. Teleport to Answer.	Apples: Get points Pink apples: Lose points Stars: Bonus Points Trees: Forest Kodu: controller	It gose for 20 Minits. you win by collecting Stars and appls and you have to go throg every stage and to win fully you have to get to the big anser at the end of the gam.

Module 4
Creating Game Worlds

Activity 4.1 Designing Game Spaces

This activity creates understanding of resource considerations, world settings and cameras. Large worlds use a great deal of resources and is crucial to use as little land as possible.

Show Tavish Hill video embedded in the course PowerPoint.

Notes

Activity 4.2 Design Patterns and Recipes

Design Patterns and Recipes allow programmers to use well-tested and understood approaches to solve programming problems. This activity will explain how game design patterns can help the game designer to use proven patterns for solving common game design problems. It will also introduce the concept of code recipes that are language specific solutions to common programming problems.

This activity will introduce the concepts of design patterns and code recipes used in the remainder of this course.

PowerPoint. Encourage the group to ask questions as they arise.

Key Points: Design patterns are standard document solutions to well known problems. Recipes are language specific code examples to solve programming problems.

Notes

Activity 4.3 Game Space Patterns

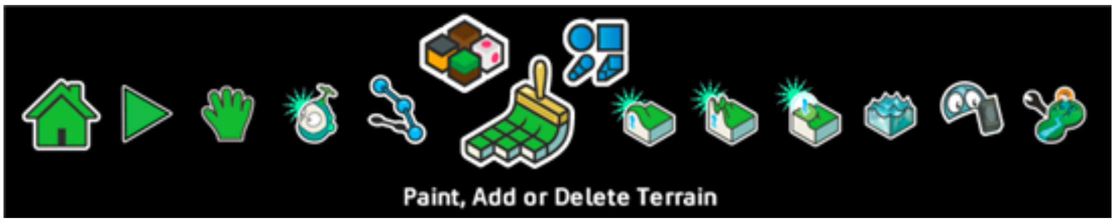
This activity will introduce some Game World Design Patterns. Each pattern has a description slide in the course PowerPoint and also has at least one demonstration game.

Design Pattern: Open World

Games using the open world design pattern allow the player to roam wherever they want. Open world games usually involve non-linear game play with players able choose the sequence in which they complete game tasks. The camera follows the player (which is the default with Kodu Game Lab) and therefore open world games are more suitable for single player games.

Key points: An open world supports open game play. Some games do not have any land. Open worlds are more suited to strategy games and games where game play is non-linear.

Notes



	Use the **Ground Brush** to add and remove ground. Hold the right trigger or the left mouse button to add ground and left trigger or right mouse button to erase ground. There are a variety of different shaped brushes and the brush size can be changed as to allow the quick addition of a large area of ground but also to make intricate designs when needed. The colour of the terrain can also be selected from a range of choices.
	Use **Up / Down Brush** to create hills and valleys. Press the button or left mouse button to smooth the selected land. The different shaped and sized brushes can be used to create different effects.
	Use the **Water Brush** to add, remove or tint water. Hold the right trigger or the left mouse button to raise the water level in your world and left trigger or right mouse button to lower the water level.
	Use the **Flatten Brush** to create flat areas and ramps. Hold the right trigger or the left mouse button to level the selected ground and the left trigger or the right mouse button to smooth the selected ground.
	Use the Roughen Brush to create bumpy areas. Hold the right trigger or the left mouse button to make the selected ground spiky and the left trigger or the right mouse button to make it hilly.
	Use the **Delete Tool** to quickly remove bots and objects from your world.

When editing a world, use F3 to turn on **Snap to Grid** feature so that the brush size stays constant, the cursor coordinates will be displayed in the bottom left hand corner of the screen.

Press L and R or the mouse scroll wheel to zoom in and out. Zooming in can be useful if your cursor is difficult to find because it is hidden behind land.

Important: When creating your worlds monitor the resource meter and ensure that it doesn't enter the red area. This is particularly important when creating large open worlds.

Design Pattern: Race

Racing games require the player to race around a pre-defined track or space. Often there are checkpoints, with the goal either to beat opponents or register a fast time. Racing games closely mirror real life races and are therefore a simple pattern to understand and to play.

Demonstrate **Pure Plasma Racing** game.

Key Points: Paths can be used to create roads for AI controlled characters.

Notes

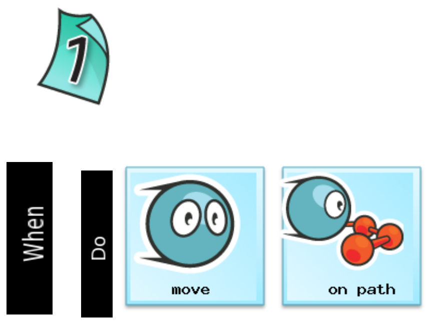
Recipe: Racing track opponents

To program bots for a player to race against simply create a path and then program the bot to follow the path.

When using the Xbox controller paths are created with the Object Tool.
When using the keyboard and mouse the Add Path button is in the tool menu

Select the Object Group and then path type you want (plain, road, wall or flora). Use the Y button to add more nodes, move the cursor to next desired position and then press A to add the node. If the race track is a loop enabling multiple laps then the last node should be placed on top of the first node. Press B to finish.

Place the bot on the path and then use the following recipe.



Design Pattern: Side Scrolling Games

A platform game is a game where the player can only move in 2D dimensions, it also called a side scroller. Side Scrolling games usually require the player to do lots of jumping, over gaps and over other obstacles that may or may not be moving. Side scrolling games were extremely popular in the 80s and have a nostalgic feel when played.

Demonstrate games Coins and Platform Cycle.

Key Points:
1. Using East / West constraint on moving

2. Fixing the camera as an offset in the World Settings (extremely useful)

Notes

____

____

____

____

____

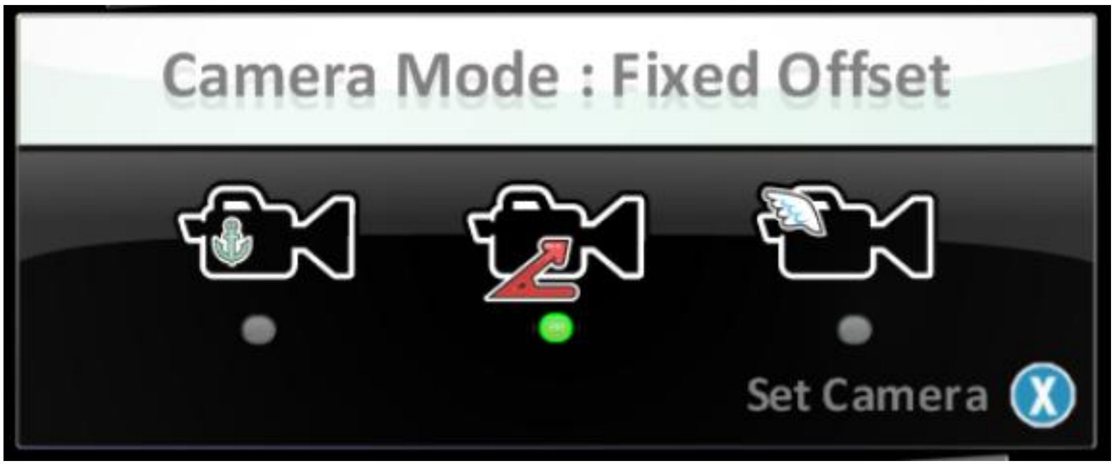
Recipe: Side Scrolling
Usually a path would be best for making a side scrolling game, but you can also create land.

We need the camera to follow the player's character from the side. To do this we will set the camera mode to fixed offset.

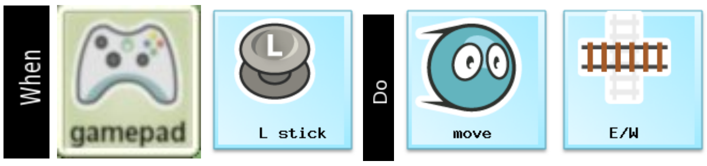
1. Click on World Settings

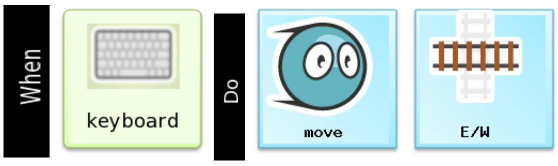
2. Select Camera Mode: Fixed setting.

3. Press the button to place the camera in its starting position and press the button to set the position.


Camera Mode : Fixed Offset

Set Camera





Design Pattern: Maze

Using a maze as your game world turns your game into a puzzle requiring the player to use skill to find the correct route. Some mazes have a single route that needs to be discovered while other mazes have multiple possible routes.

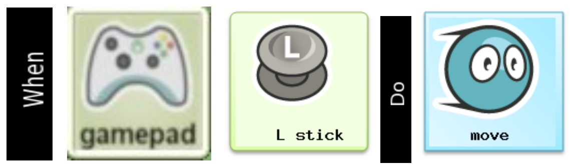
Key points:
1. Demonstrate camera in first person position

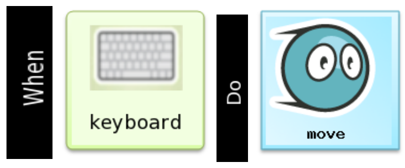
Notes

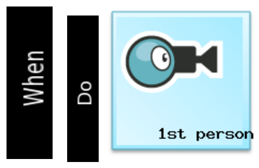
Recipe: Maze Game

When making a maze it is easier to start by raising the entire maze area and then lowering the maze between the walls. Raising the maze walls is much harder.

For a maze game it is useful to either set the camera as a fixed offset (as for the side scrolling game) behind the player's bot or alternatively set the camera to first person. To set the camera to first person use the following recipe.


When gamepad L stick Do move


When keyboard Do move


When Do 1st person

Design Pattern: Fixed Game Board

Some games spaces are small fixed areas. These game spaces are more suited to multiplayer games without split screen functionality. By having a set space and a fixed camera all players have the same perspective.

Games such as PacKodu are good example. Key points: Suitable for multi-player games

Notes

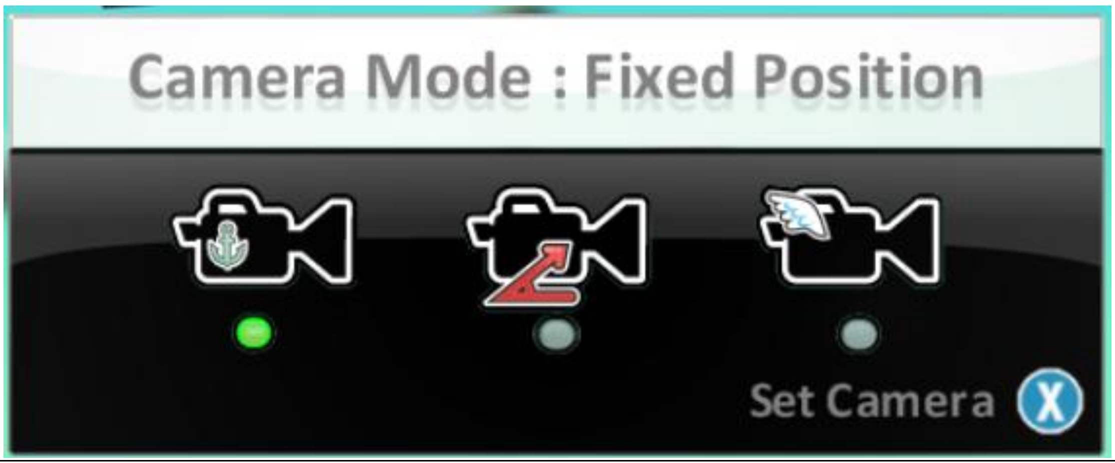
Recipe: Fixed Game Board

We need the camera to stay in the same position for the entire game. To do this we will set the camera mode to fixed position.

1. Click on World Settings

2. Select Camera Mode: Fixed Position.

3. Press the X button to place the camera in its starting position and press the A button to set the position.



Camera Mode : Fixed Position

Module 5
Game Design Patterns

Activity 5.1 Game Progress Patterns

This activity introduces advanced Kodu programming concepts, such as creatables, pages and using scores as variables.

Design Pattern: Time Limits
Time Limits require a player to complete an action or achieve a goal or alternatively sets a time that the player must survive in the game in order to win. Countdown clocks usually display the time remaining to give the game a sense of urgency. Some games feature time bonuses that are gained through achieving certain tasks.

Notes

___

___

___

___

___

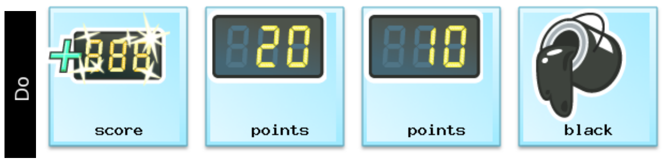
Kode Recipe: Time Limit

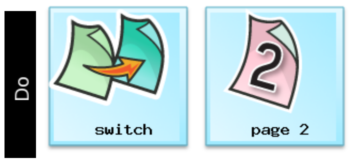
Notes:
1. Make sure you attach the game timer to a bot or object that cannot be destroyed during the game.

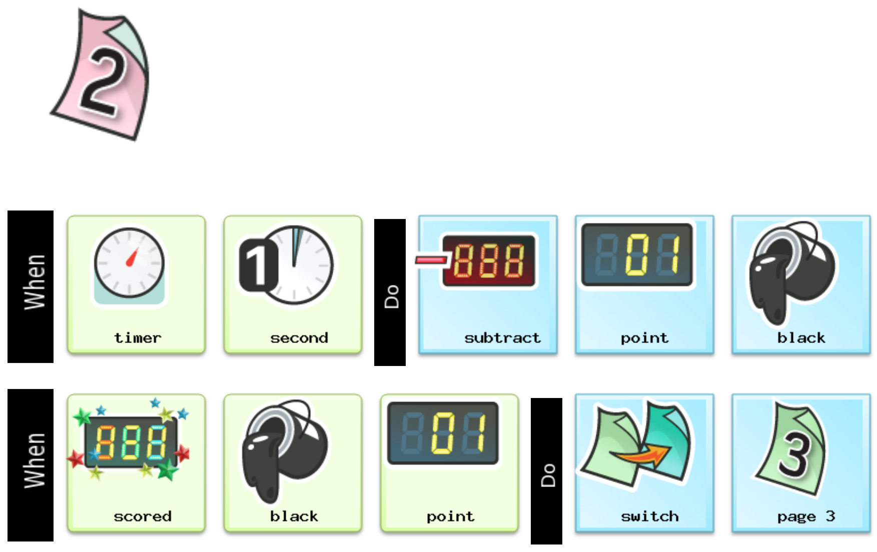
2. Select a color to use as your timer and do not use the color score for anything else.

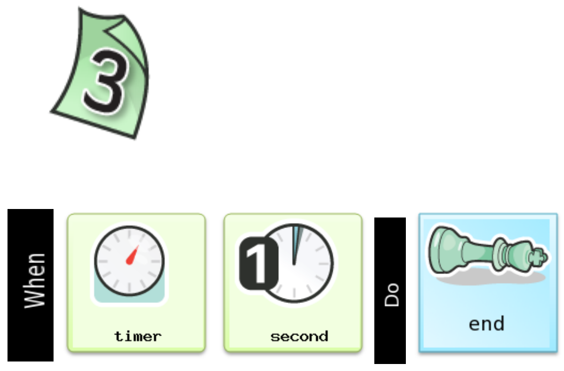
3. This example recipe sets the time for 30 seconds and uses the colour black for the timer. After 30 seconds the player wins the game.

4. Refer to the game Wack-a-Lama to see this recipe in action.









Design Pattern: Scores

Scores are a numerical representation of a player's success. Points can be added to a players score for achieving certain goals or deducted for failed activities. A player's score is usually displayed at all times during a game, often points achieved are displayed as an overlay as the goal is achieved adding to the sense of achievement and progress.

Notes

Kode Recipe: Scores

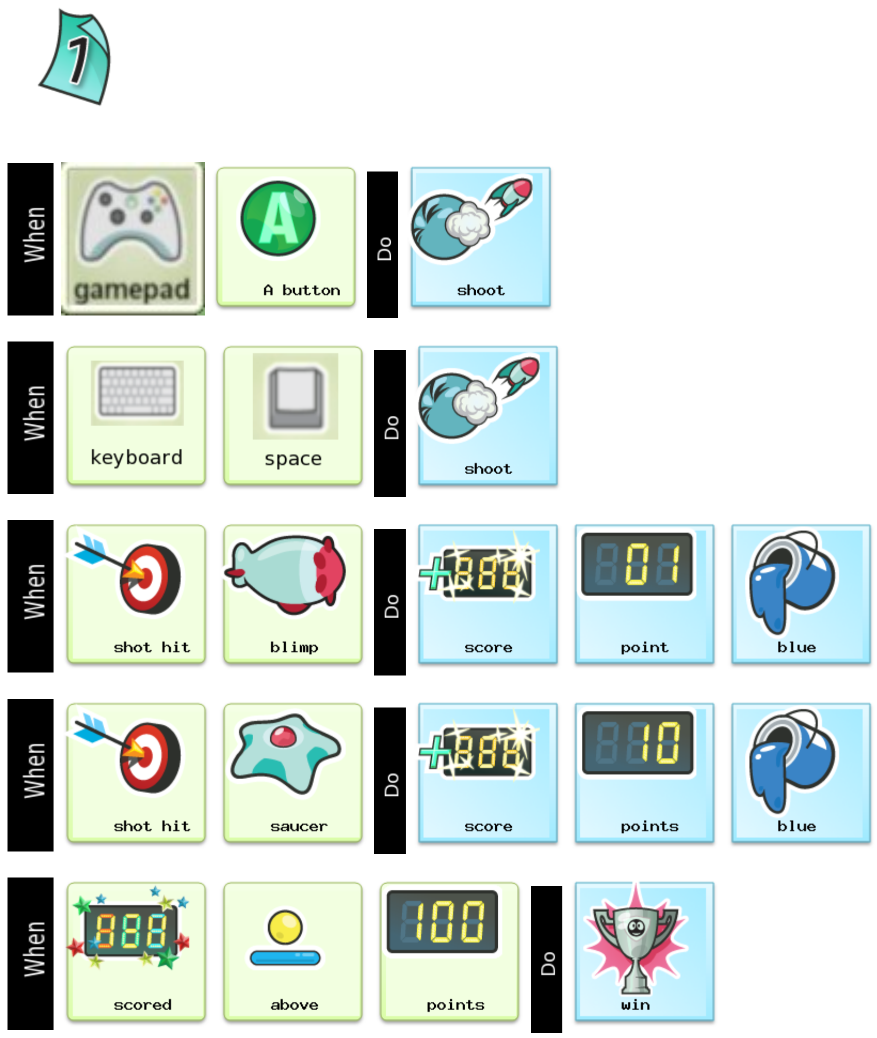
Notes:

1. This recipe adds 1 point to the blue score each time a shot hits the blimp and 10 points each time a saucer is hit. When the score is more than 100 the player wins the game.

2. Add this recipe to the bot that is doing the shooting.

3. This recipe works with both the keyboard and the gamepad.

4. Refer to Xevon v06 to see this recipe.



```
1
When gamepad A button Do shoot
When keyboard space Do shoot
When shot hit blimp Do score point blue
When shot hit saucer Do score points blue
When scored above points Do win
```

Pattern: Health

The health of the players bot can also be used to indicate progress within the game. A health bar shows the current health of the character and provides immediate feedback to player.

Notes

Recipe: Health

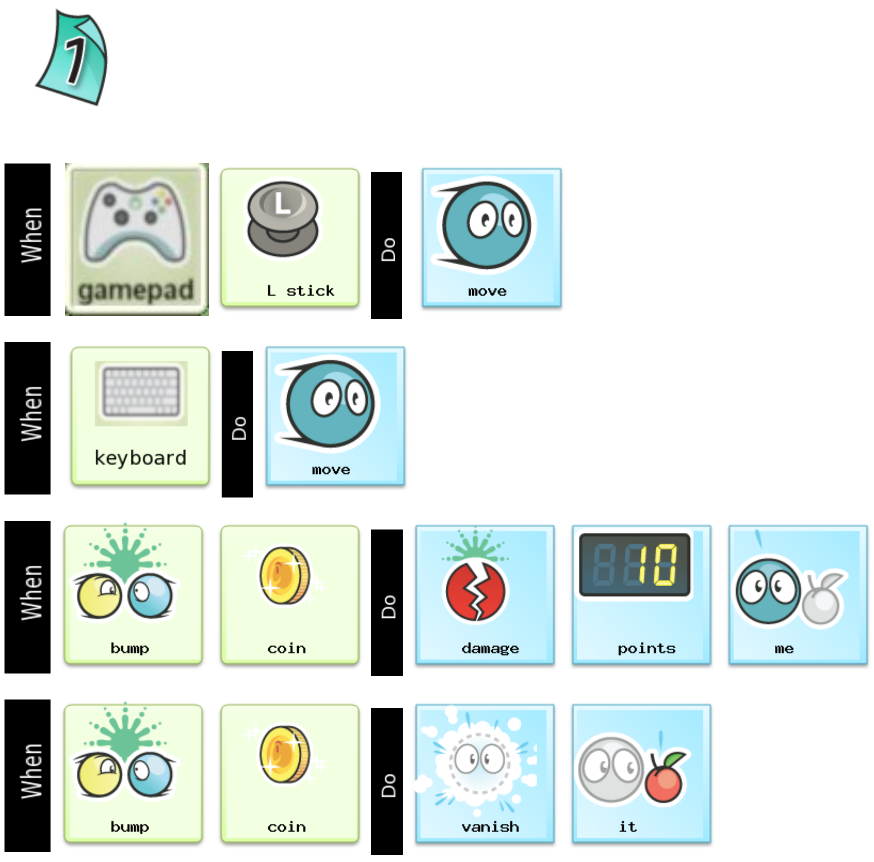
Notes:

1. When the bot touches a coin 10 points is deducted from the bots health and the coin vanishes.

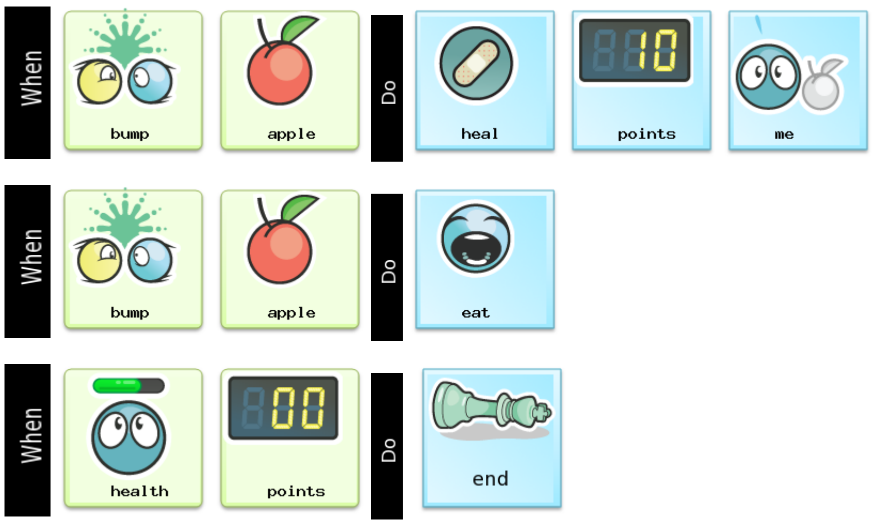
2. When the bot touches and eats an apple 10 points are added to the bots health.

3. When the bot's health is zero the game ends.

4. The number of Hit Points (health) a bot starts with is set in the bot's settings. Values between 0 and 1000 in increments of 5 can be set.



```
When gamepad L stick Do move
When keyboard Do move
When bump coin Do damage points me
When bump coin Do vanish it
```



Pattern: Save Points

Save Points (also called Check Points) are convenient points in a game, usually after a hard section of the game has been completed where the game is either saved automatically or the player is given the option to save given the option. If the user fails during the next section of the game, the game is restarted from the save point rather than returning the player to the beginning of the game. This alleviates the problem of players needing to replay easier sections of the game in order to reach their sticking point.

Notes

Recipe: Save Points

Note:

1. The cycle bot, which is the player controlled character, is a creatable.

2. The red score is used to keep track of the players lives. A value of 1 indicates that the player is alive and playing while a value of 0 indicates that the player's bot needs to be respawned.

3. The black score is used to keep track of the save points. 0 indicates the player is at the beginning of the game. 1 indicates that the player has reached save point 1, and so on.

4. A bot has to be created and programmed at each save point, in this example we've used stick bots but the type of bot doesn't matter.

5. There isn't any code for page 3 of the save points.

6. Use the settings of the save points to reduce the hearing to its minimum value so that the stick bot only hears the cycle bot when it is very close.

7. A different colour score could be used to track and limit the number of new lives, this is not shown in the recipe.

The player controlled bot is a creatable

1

When Do score red point once

When gamepad L stick Do move

When keyboard Do move

When bump apple Do subtract red point

{indent} When Do boom me

Starting point (save point 0)

1

When Do switch page 2

2

When scored red points Do

{indent} When timer second Do create cycle

When scored black point Do switch page 3

Save Point 1

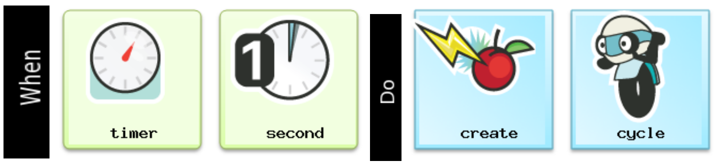
1

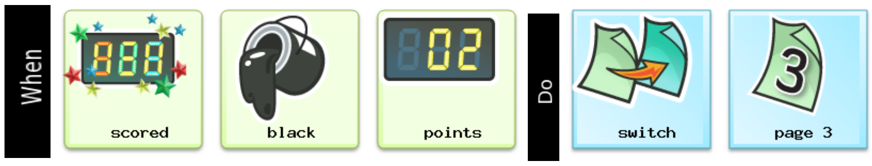
When hear cycle Do score point black once

{indent} Do switch page 2

2

When scored red points

{indent} 



Successive save point will need their black points incremented by 1

Design Pattern: Multiple Levels

Having multiple levels in a game is a great way to convey a sense of progress to the player. Games also use levels to increase the difficulty and/or introduce new game mechanics.

A great example of multiple levels in use is halox's Dual and Portal games, both of these are available as youtube videos.

Key points: This is an advanced example not for the faint hearted!

Notes

Recipe: Multiple Levels

Notes:

1. The black scores are used to keep track of which level the player is currently on.

2. The player's bot is a creatable and therefore must be created at the start of the game.

3. Subsequent levels can be made by copying the Level 2 start and end points and incrementing the black scores in the code.

4. Set the black score to **Off**

The player controlled bot is a creatable

When gamepad L stick Do move

When keyboard Do move

Starting point on level 1

Do create cycle once

Finishing point on level 1

When bump cycle Do vanish it

{indent} Do score point black once

Stickbot: Starting point on level 2

When scored black point Do create cycle once

Stickbot: Finishing point on level 2

When bump cycle Do vanish it

{indent} Do score points black once

Activity 5.2 Game Play Patterns

Pattern: Power Ups

Power-Ups give a time limited advantage to the player that activates them. Power Ups require players to make strategic decisions about when to use them as power up need to be earned or require time to recharge.

Power Ups solve the problem of having to complete a variety of tasks of varying difficultly as they let the user have increased abilities to face harder challenges without making the tasks (without power ups) too easy.

For example, a character may become invulnerable or be able to move faster for a period after obtaining an item or after a certain key is pressed. Sounds or visual effects can be used to signal that are power up is being used and/or about to end.

Key Point: This recipe shows the use of pages and a timer.

Notes

___

___

___

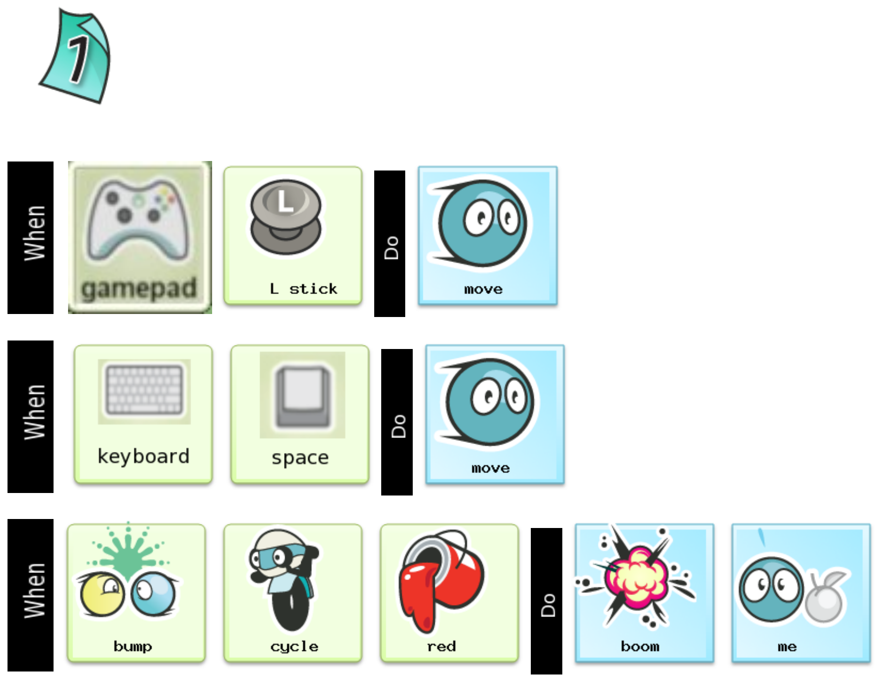
___

___

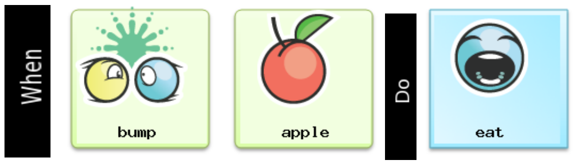
Recipe: Power Up

Notes: This recipe programs a bot which is controlled by a player to lose the game when it is bumped into by a red cyclebot. When the player's bot bumps and eats an apple it is powered up so that the red cycle bots are now destroyed when they are touched by the players character. The power up lasts for 5 seconds and then the player's bot returns to its original state.

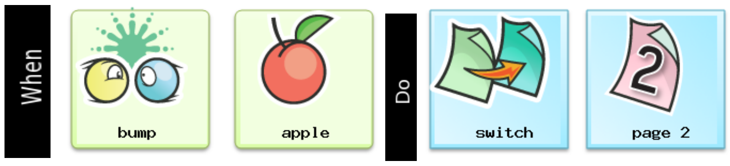
1. Notice the use of me and it.



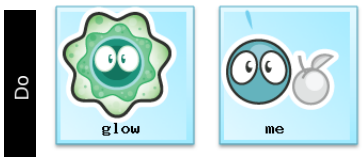
```
1
When gamepad L stick Do move
When keyboard space Do move
When bump cycle red Do boom me
```



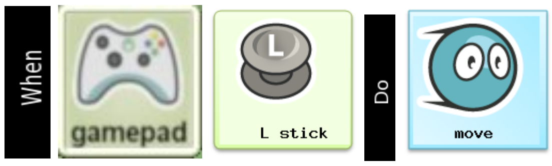
```
When bump apple Do eat
```



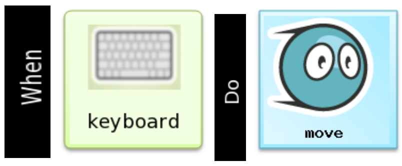
```
When bump apple Do switch page 2
```



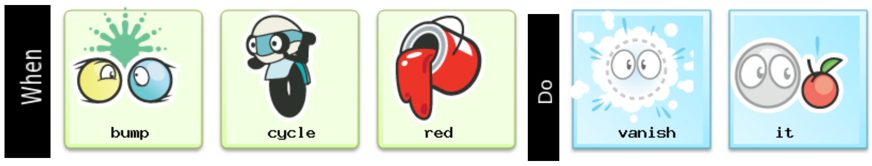
```
Do glow me
```



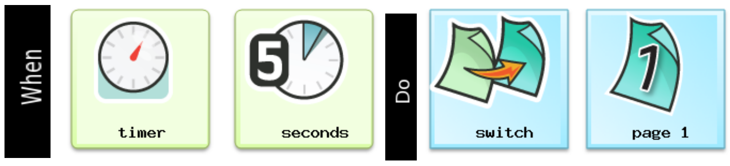
```
When gamepad L stick Do move
```



```
When keyboard Do move
```



```
When bump cycle red Do vanish it
```



```
When timer 5 seconds Do switch page 1
```

Pattern: Transfer of Control

Some games allow the player to control different characters at different stages of the game. This could include when a player's character enters a car or boards a boat. The new character would usually have different abilities and therefore the game play would be different resulting in greater interest for the player.

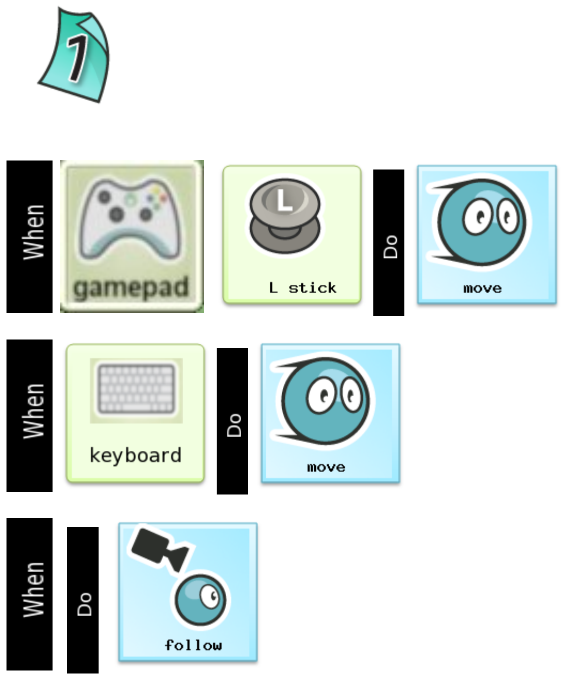
Notes

Recipe: Transfer of Control

Notes:

1. This example shows how a player controlled cycle bot can "get into a boat" which the player then controls.

Cyclebot: Initial player controlled bot

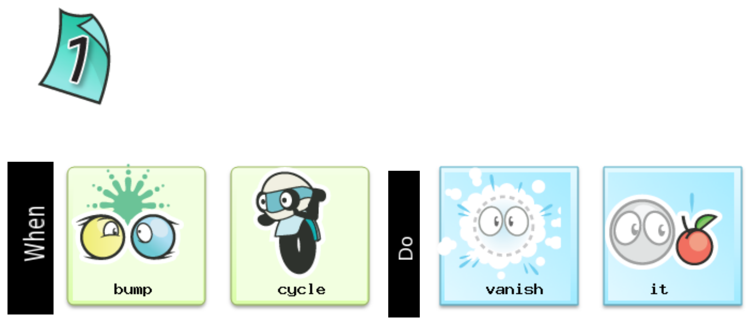


When gamepad L stick Do move

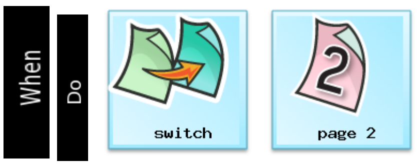
When keyboard Do move

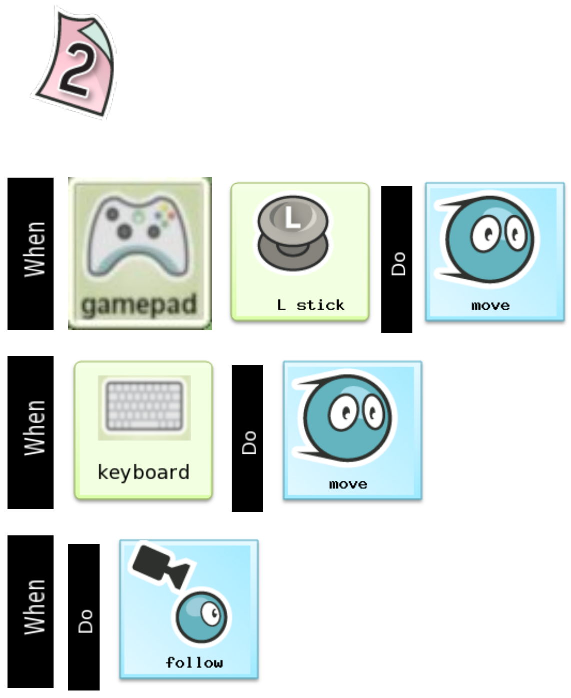
When Do follow

Floatbot: Player controlled bot once the cyclebot has boarded it.



{indent}





Pattern: Big Boss

Games or levels in games often finish with a battle with a Big Boss. The Big Boss is a much more difficult opponent and often has different abilities to the previous opponents. Battling the Big Boss is a great way to build a sense of progress and to give closure to game.

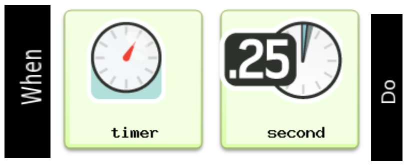
Notes

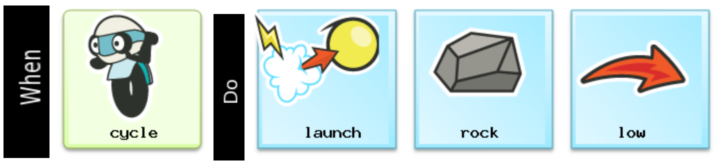
Recipe: Big Boss

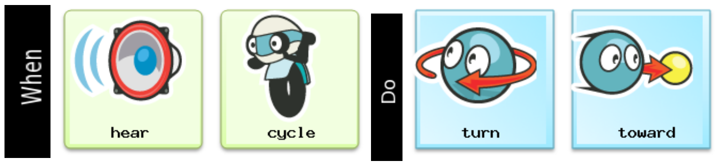
Notes:
1. The bots hit points settings for a Big Boss can be increase and therefore made harder to defeat.

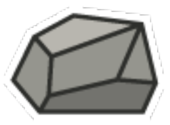
2. The bots damage points can be increased to make the attacks more damaging.

3. This only one example of how a big boss could be made.

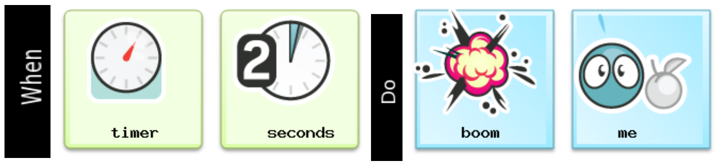


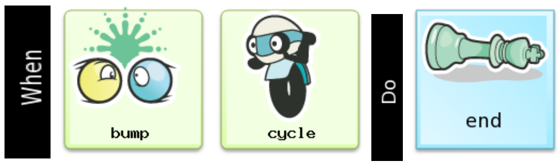
{indent}


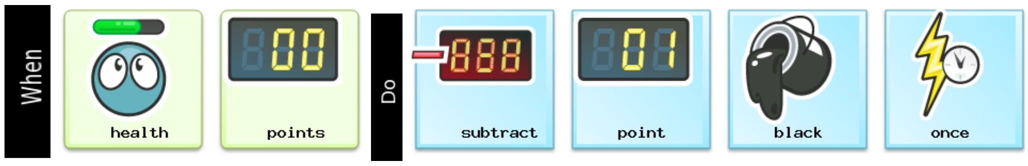




Rocks are thrown by the Boss, so create a single creatable rock.







Activity 5.3 Game Information Design Patterns

It is important to communicate with players so that they clearly understand the state of the game and their progress within it. Game information can be communicated with text, sounds, scores and change to the game space. Clear communication helps prevent frustration and unexpected outcomes for the player.

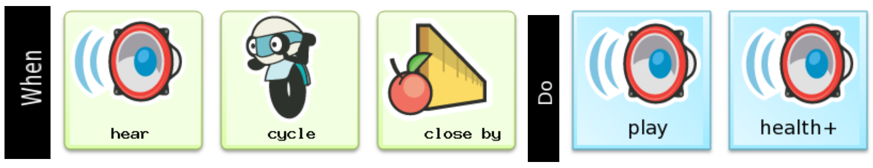
Pattern: Alarm

Alarms can be used in games to warn players of danger. They can be used indicate that a phase of the game is about to begin or end. Different sounding alarms can be used to notify the player of different events.

Notes

Recipe: Alarm

This recipe shows how an alarm can be used to indicate that cycle bot is close by.



Pattern: In Game Information

Dialog boxes can be used to display information to the user, this information may simply notify the user of their progress in the game or it may provide information that the player needs to successfully complete the game.

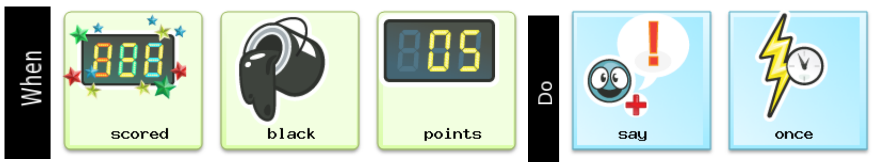
Notes

Recipe: Dialog

There are three kinds of dialog boxes in Kodu Game Lab: Full Screen and two Thought Balloons, one in which the lines are displayed sequentially and one in which lines are displayed randomly.

Full screen is used to give general messages to the player, while speech dialog boxes are used when specific characters are used provide information to the player.

In this example a full screen dialog will be displayed when the black score reaches five points.



Use the Y button once you selected the say icon to select which type of dialog will be used.

Module 6
A Game Design Approach

Activity 6.1

In this example we will investigate design processes suitable for use by students.

Jesse Schell says, “think of an problem, identify the best way to solve the problem, implement, test.”

Some advice:

Plan your game on paper.

Start with simple game play and then expand it.

Test, test and test.

Describe the design process that you plan to use with your students

Activity 6.2

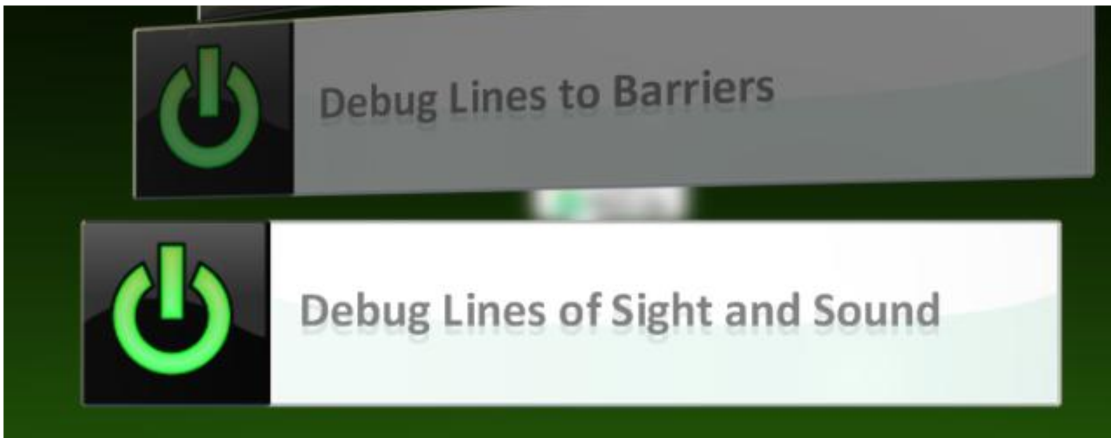
In this activity we will investigate some of the functionality in Kodu Game Lab that makes life easier for game designers.

Identifying risks: Can we do this in Kodu?

There are many things that are possible or impossible to create with Kodu Game Lab. When designing a new game it is crucial that the game designer identifies the parts of the game that may not be possible and creates them first, that way if the game is impossible to create time has been wasted.

Debugging: Why isn’t this working?

When the game is not playing as expected there are debugging lines that can be turned on. As the name suggests the Debugging Lines of Sights and Sounds will show where the bot can and can’t see and hear and is very useful in discovering why any code is not working as expected.


Debug Lines to Barriers

Debug Lines of Sight and Sound

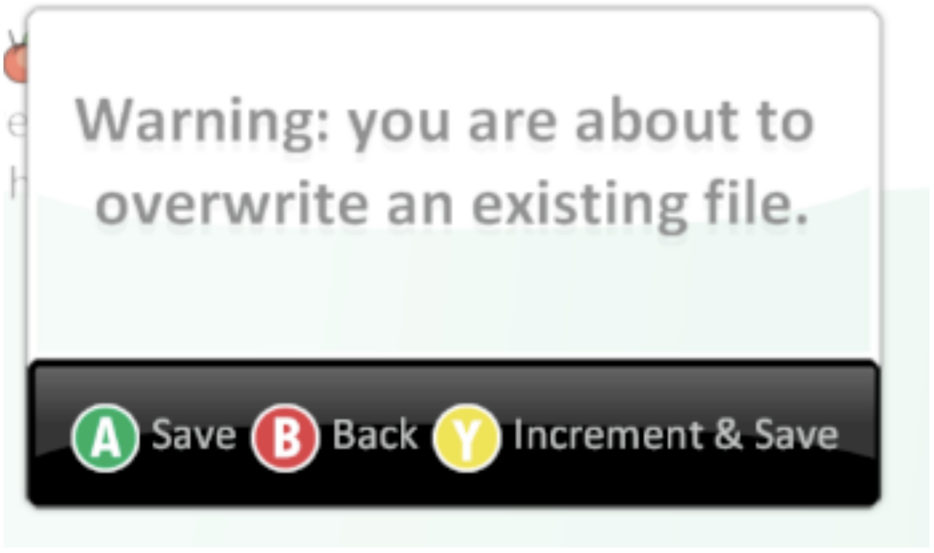
Copy, Paste, Clone: Is there a quicker way to do this?

When creating multiple bots with the same programming it is often quicker and easier to clone and paste the bot or object. Bots and objects can also be cut and pasted between projects.

Use the L to cut and R to clone and paste or alternatively right click the object with the mouse.

Using Versioning: Whoops my game no longer works!?

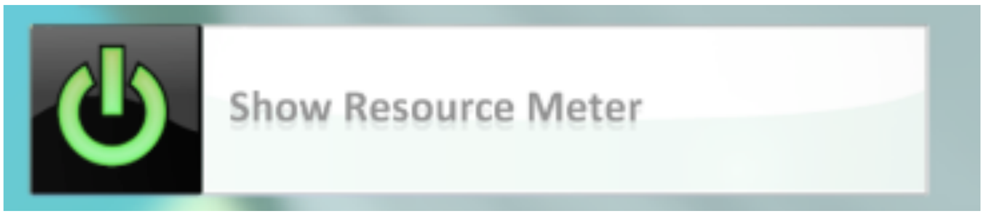
When saving major changes to Kodu Game Lab games it useful to save it as a new version. This effectively creates backups of your game which is useful if you happen to make changes that cause your game to stop working. When saving a new version of a game Kodu will warn you and suggest that you save it as an increment, press Y to do this.


Warning: you are about to overwrite an existing file.

Using the Resource Meter: Why is my game running so slowly?

Kodu Game Lab is for making small games and when games get too large Kodu starts to slow down, often making the game unenjoyable and unplayable.

The resource meter can be displayed by turning it on in the World Settings.



Ensure the resource meter stays out of the red while testing your game. Some times your code can be modified to limit the resources used eg by minimising hear commands but usually it will be necessary to reduce the amount of land used in the game.

Module 7
Next Steps

Discuss with the group how schools have used Kodu Game Lab and how they intend to introduce it at their schools.

How do you plan to use Kodu Game Lab at your school?

Kodu Resources

Websites

http://fuse.microsoft.com/kodu

http://planetkodu.com

http://boards.kodux.com

All workshop materials are downloadable from

http://media.planetkodu.com/workshop/resources.html