PRODUCT DESIGN FOLIO

Stage 2 DTE: Game Development

Word Count: 1823 + 1-minute video (166 words) = 1989 Words Total

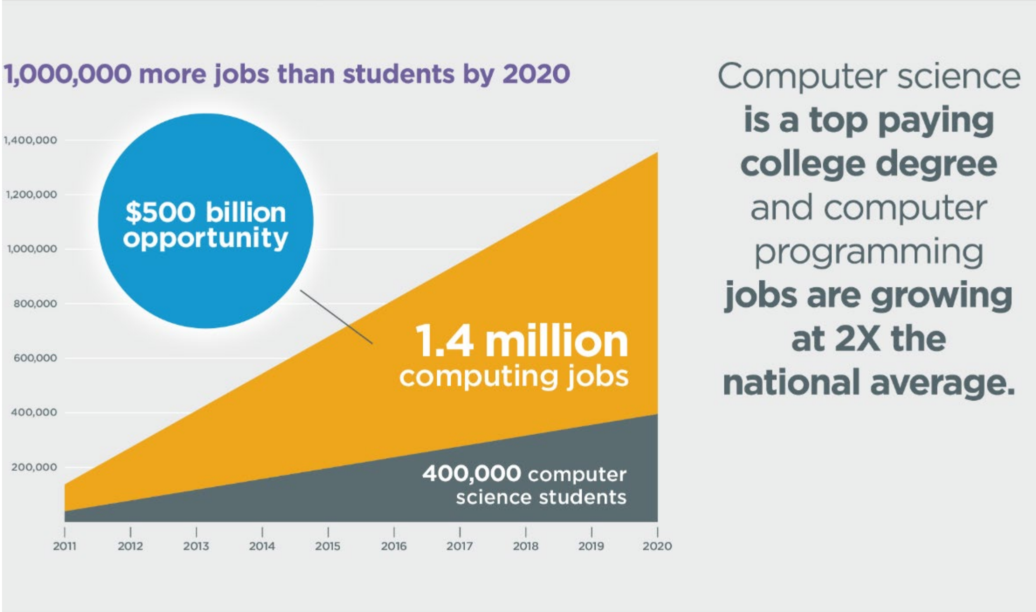
Please watch the attached ProductEvidenceVideo
For walkthrough/evidence of product
(not part of Folio)

STATEMENT OF INTENT

The demand for employees with coding skills has been steadily increasing and shows no signs of slowing down. The average rate of growth for all jobs by 2028 is expected to be 5%, Software Development jobs are expected to grow 21%. Now more than ever it's important to instil a love of computing and coding in young children and inspire them to explore this as a career path.

I intend to create a game-based application that teaches junior primary children how to code utilising a block-based programming language to complete various tasks. This will teach them the basics of programming as they learn coding fundamentals. These tasks will be spread out around a semi-open world isometric game.

Importance of coding skills:



Coding jobs are in an enormous deficit and showing no signs of slowing down. Learning coding skills not only prepares children for a future in the sector, but also in any future career, with employees valuing coding skills, especially in high paying jobs.

A product that teaches these skills would clearly be well received by both parents; due to future opportunities it grants their children.

MIND MAP – EDUCATIONAL CODING GAME

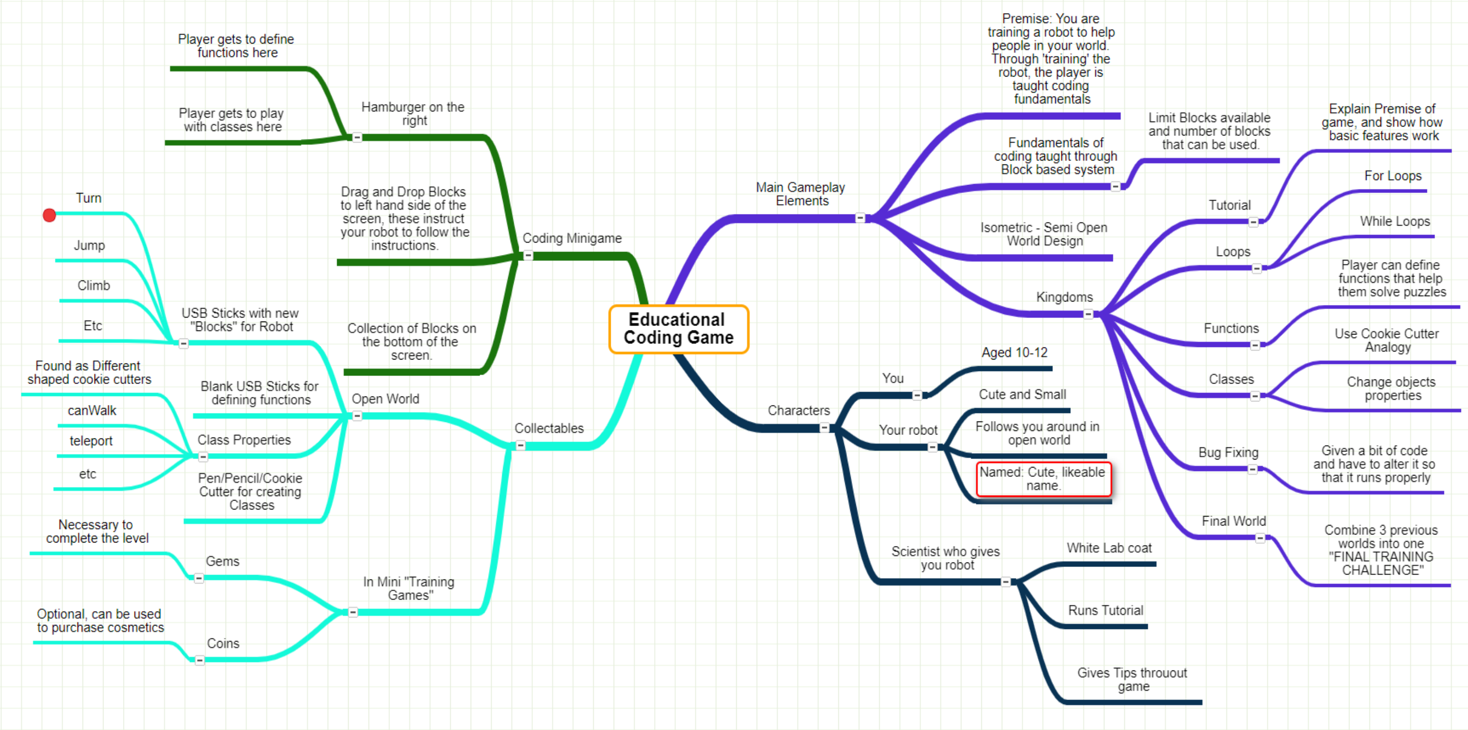
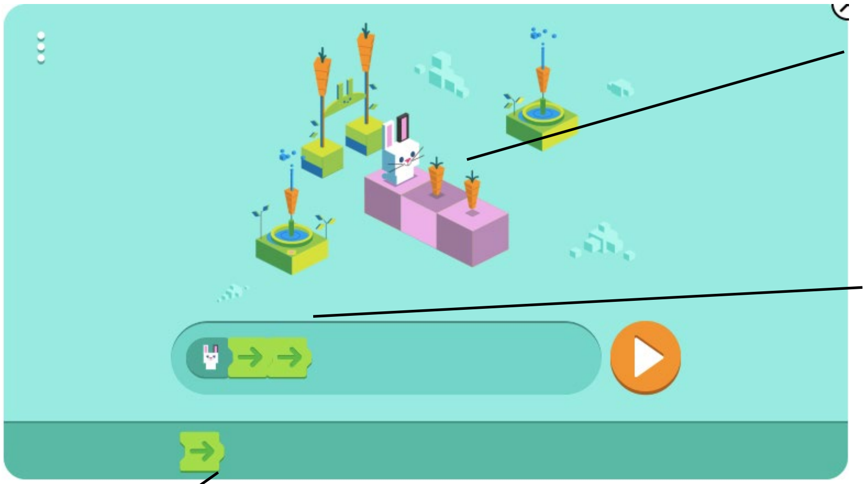

Educational Coding Game
Coding Minigame
Hamburger on the right
Player gets to define functions here
Player gets to play with classes here
Drag and Drop Blocks to left hand side of the screen, these instruct your robot to follow the instructions.
Collection of Blocks on the bottom of the screen.
Collectables
Open World
USB Sticks with new "Blocks" for Robot
Turn
Jump
Climb
Etc
Blank USB Sticks for defining functions
Class Properties
Found as Different shaped cookie cutters
canWalk
teleport
etc
Pen/Pencil/Cookie Cutter for creating Classes
In Mini "Training Games"
Gems
Necessary to complete the level
Coins
Optional, can be used to purchase cosmetics
Main Gameplay Elements
Premise: You are training a robot to help people in your world. Through 'training' the robot, the player is taught coding fundamentals
Fundamentals of coding taught through Block based system
Limit Blocks available and number of blocks that can be used.
Isometric - Semi Open World Design
Kingdoms
Tutorial
Explain Premise of game, and show how basic features work
Loops
For Loops
While Loops
Functions
Player can define functions that help them solve puzzles
Classes
Use Cookie Cutter Analogy
Change objects properties
Bug Fixing
Given a bit of code and have to alter it so that it runs properly
Final World
Combine 3 previous worlds into one "FINAL TRAINING CHALLENGE"
Characters
You
Aged 10-12
Your robot
Cute and Small
Follows you around in open world
Named: Cute, likeable name.
Scientist who gives you robot
White Lab coat
Runs Tutorial
Gives Tips throuout game

Figure 1: A Mind Map showing possible elements of the game

Analysis: Branching out from my essential idea, I confirmed the finer details of my game, including characters, collectables, and gameplay elements.

EXISTING PRODUCT ANALYSIS

Product 1: Google Doodle Celebrating 50 years of Kids Coding: Web Game



Isometric Design, each block alternates colours, to make it clear how many blocks there are.

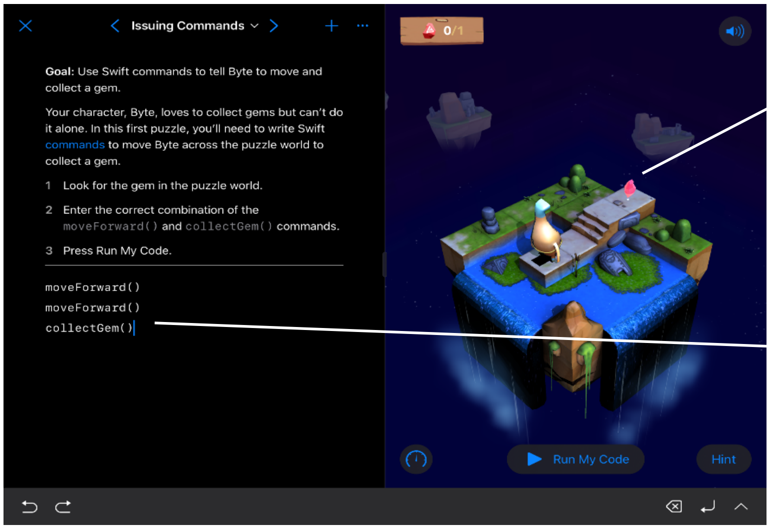
Horizontal assortment of blocks. This goes against the vertical nature of coding pieces click into each other.

Blocks have no words – simple to understand

Figure 2: In Game Screenshot of the Google Doodle designed to help kids learn to code, with analysis of its features

Analysis: Its isometric design indicates the number of blocks needed to travel. Personally, code running vertically is more intuitive than horizontally. Pictures on blocks and the blocks "clicking" together, like a jigsaw, as works well with a younger target audience.

Product 2: Swift Playgrounds: App



Goal: Use Swift commands to tell Byte to move and collect a gem.

Your character, Byte, loves to collect gems but can't do it alone. In this first puzzle, you'll need to write Swift commands to move Byte across the puzzle world to collect a gem.

1 Look for the gem in the puzzle world.

2 Enter the correct combination of the `moveForward()` and `collectGem()` commands.

3 Press Run My Code.

```
moveForward()
moveForward()
collectGem()
```

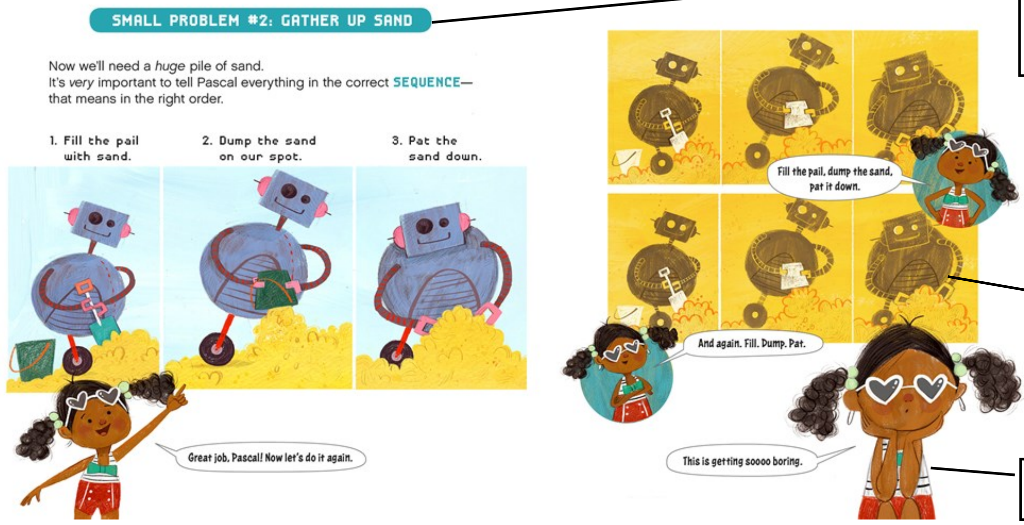
Isometric design, position of blocks less clear as constant colours

Code order vertically, but not blocks, words.

Figure 3: A Screenshot of Apples "Swift Playground" with annotations attached

Analysis: The isometric design again works well, however the number of blocks between the player and gem is unclear, as shown by my code, I only moved forward twice, when 3 moveForwards() were necessary. This is produced by Apple, with the intention of getting children to learn their proprietary coding language and is aimed at an older audience.

Product 3: How to Code a Sandcastle: Children's Book



SMALL PROBLEM #2: GATHER UP SAND

Now we'll need a *huge* pile of sand. It's *very* important to tell Pascal everything in the correct **SEQUENCE**—that means in the right order.

1. Fill the pail with sand.
2. Dump the sand on our spot.
3. Pat the sand down.

Great job, Pascal! Now let's do it again.
Fill the pail, dump the sand, pat it down.
And again. Fill. Dump. Pat.
This is getting soooo boring.

Teaches kids to break problems down into smaller problems

"Hand Drawn" art style, works very well in products for children

Allusion to Loops

Figure 4: An Excerpt of the children's book "How to Code a Sandcastle"

Analysis: I like the "Hand Drawn" art style, and this will be considered for my product. As a book, it works well to teach kids how to solve problems, with coding as a secondary aspect.

DESIGN BRIEF OUTCOME CRITERIA

Criterion 1:

- Open world 2D Game
    - Character with robot that follows you around
    - Interactable chests with collectables inside
    - Gates that open by solving problems

Criterion 2:

- Coding Minigames
    - Moveable blocks controlling robot
    - Simple “For Loop” blocks
    - Function Definitions
    - Bug fixing
    - Classes

Criterion 3

- Minigame Start and End Screen
    - Start screen showing settings
        - Sound on/off
    - End Screen
        - If successful, show how many code pieces they used vs minimum
        - Retry option
        - Continue option
- Start and End screen for Main game
    - Scientist giving you / retrieving the robot.

Criterion 4

- Multiple Levels
    - Multiple “kingdoms” for each coding fundamental
    - ~5 minigame levels per kingdom

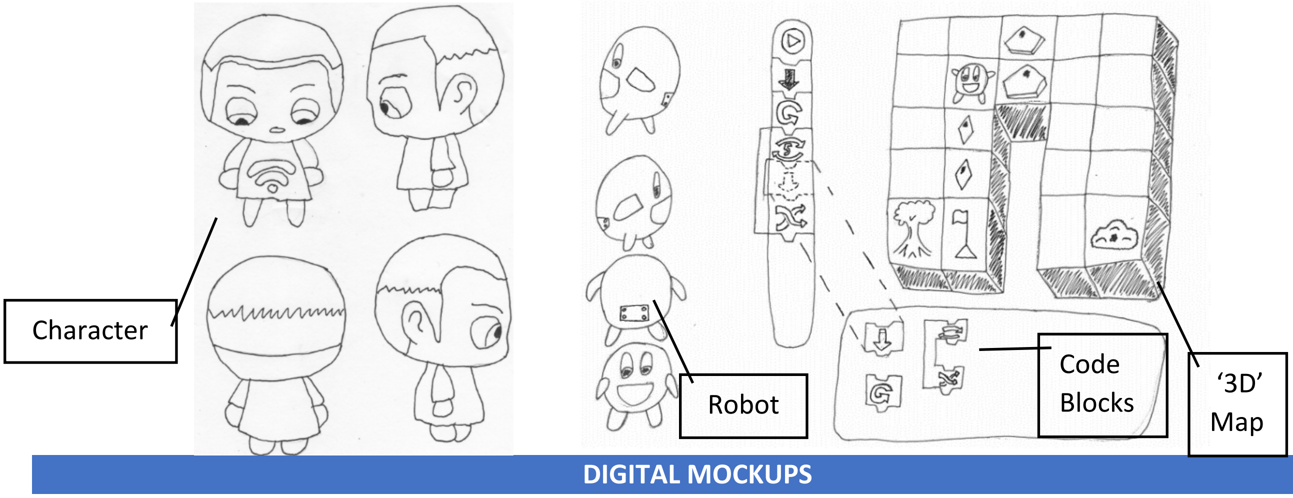
Criterion 5

- Sound Effects
    - Ambient Background Noise
    - Sound effects
    - Voice Overs

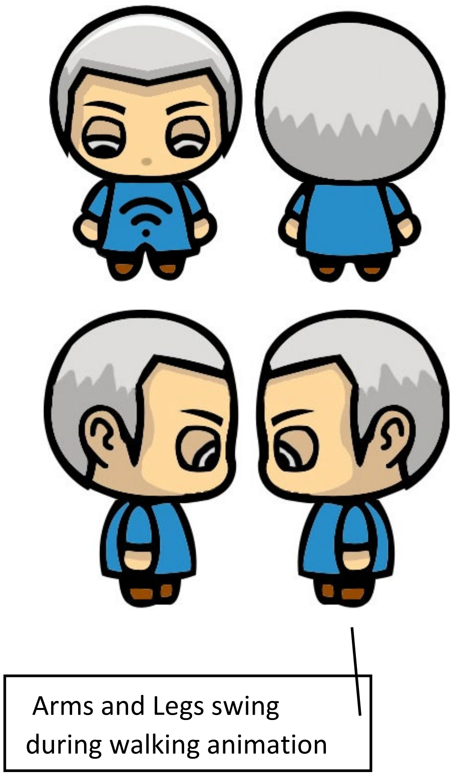
Addition / Modified Criterion 1 – Custom Level Generator

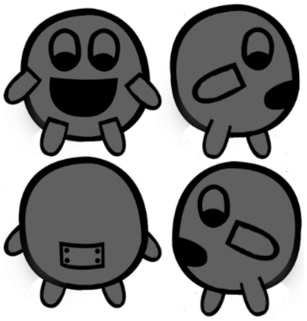
- Ability to design, play and export a custom-made level

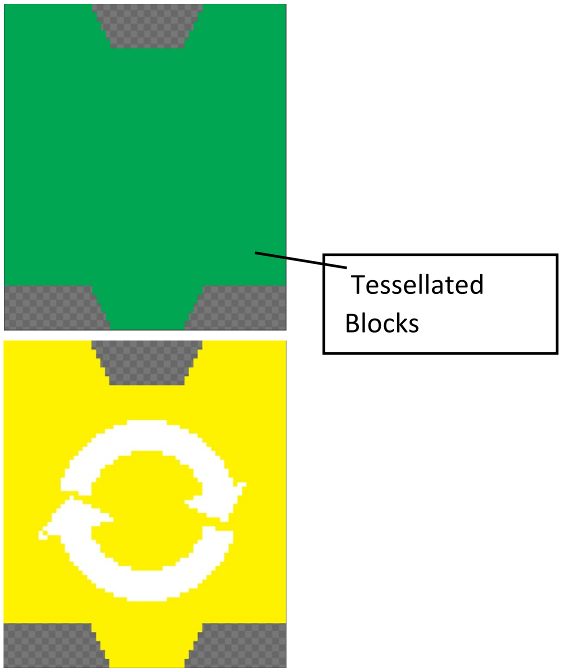
PLANNING
SKETCHING



DIGITAL MOCKUPS





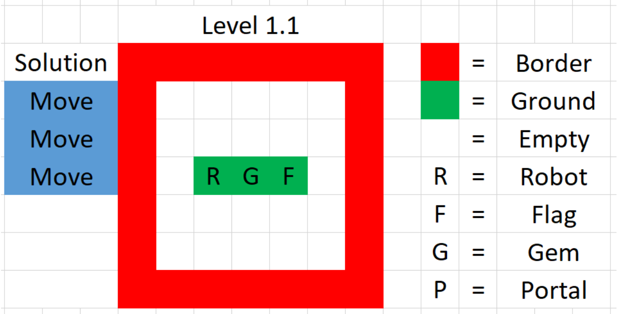


Analysis:

Photoshop and Illustrator were the main graphic design tools utilised. The tesellated blocks were created in the gamemaker sprite editor, as it was simpler to create pixel-perfect artwork as necessary for the blocks to tessellate. Creating a 3D map is going to be challenging, as gamemaker only allows 2D so blocks would have to be layered in the correct fashion, as a fallback a 2D top down art set will be created.

LEVEL DESIGN PLANNING

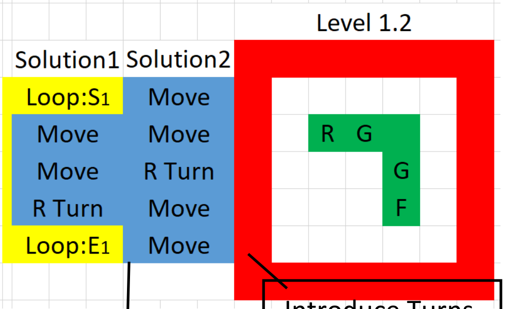
Levels were designed in excel: the grid nature made designing simple, and drawing sketches was unnecessary.



Solution
Move
Move
Move

🟥	=	Border
🟩	=	Ground
	=	Empty
R	=	Robot
F	=	Flag
G	=	Gem
P	=	Portal

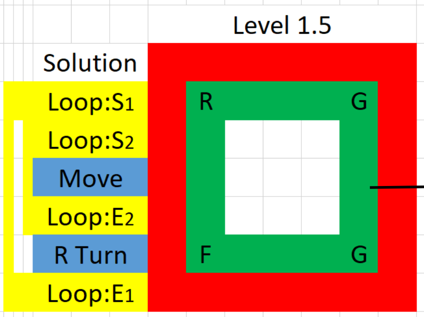
First Level, introduce Drag and Drop and Moving



Solution1	Solution2
Loop:S1	Move
Move	Move
Move	R Turn
R Turn	Move
Loop:E1	Move

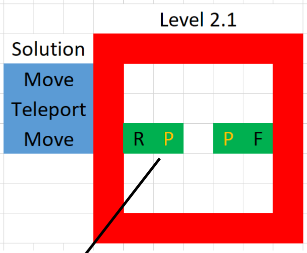
Introduce Turns

Two possible solutions of equal length if Loops are allowed. A system will be made to only allow only select blocks to be used.



Solution
Loop:S1
Loop:S2
Move
Loop:E2
R Turn
Loop:E1

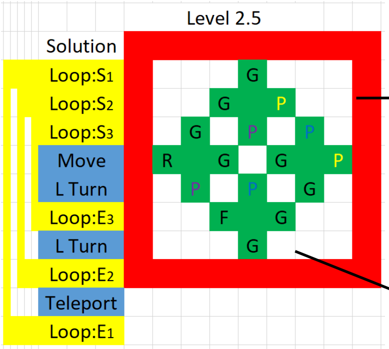
Final Level in World 1. Testing their knowledge of Loops using nested Loops.



Solution
Move
Teleport
Move

Text Colour of portal indicates which portal it links to

World 2 introduces portals



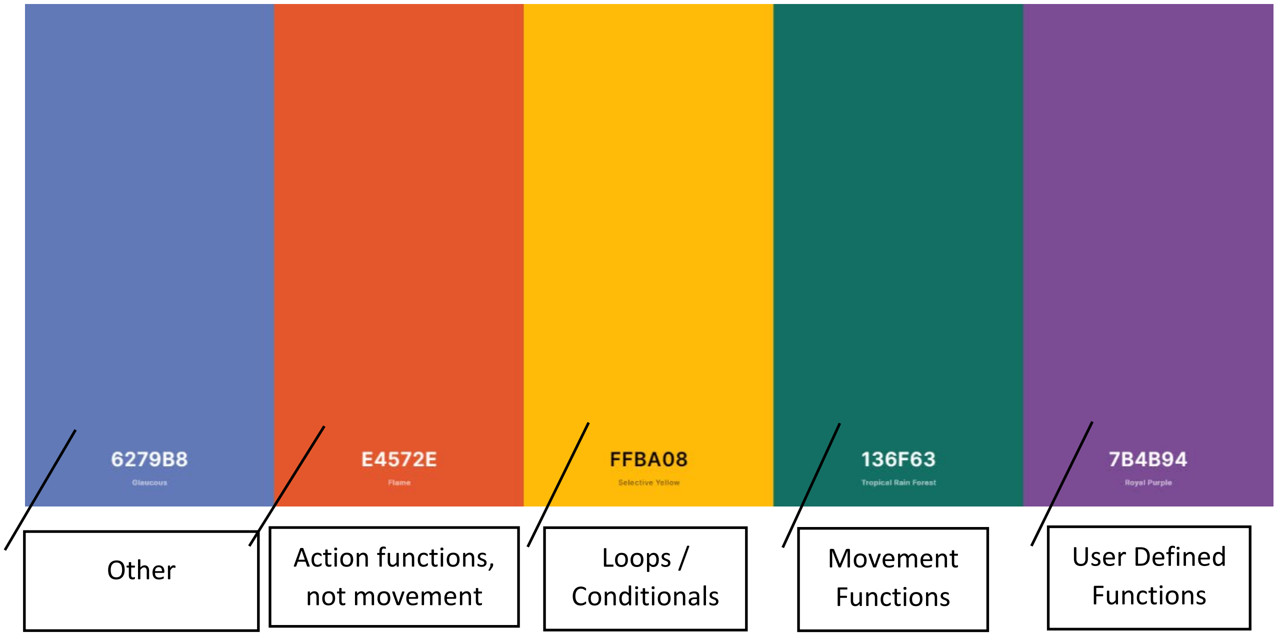
Solution
Loop:S1
Loop:S2
Loop:S3
Move
L Turn
Loop:E3
L Turn
Loop:E2
Teleport
Loop:E1

Border size increased – possible because of my scalable system.

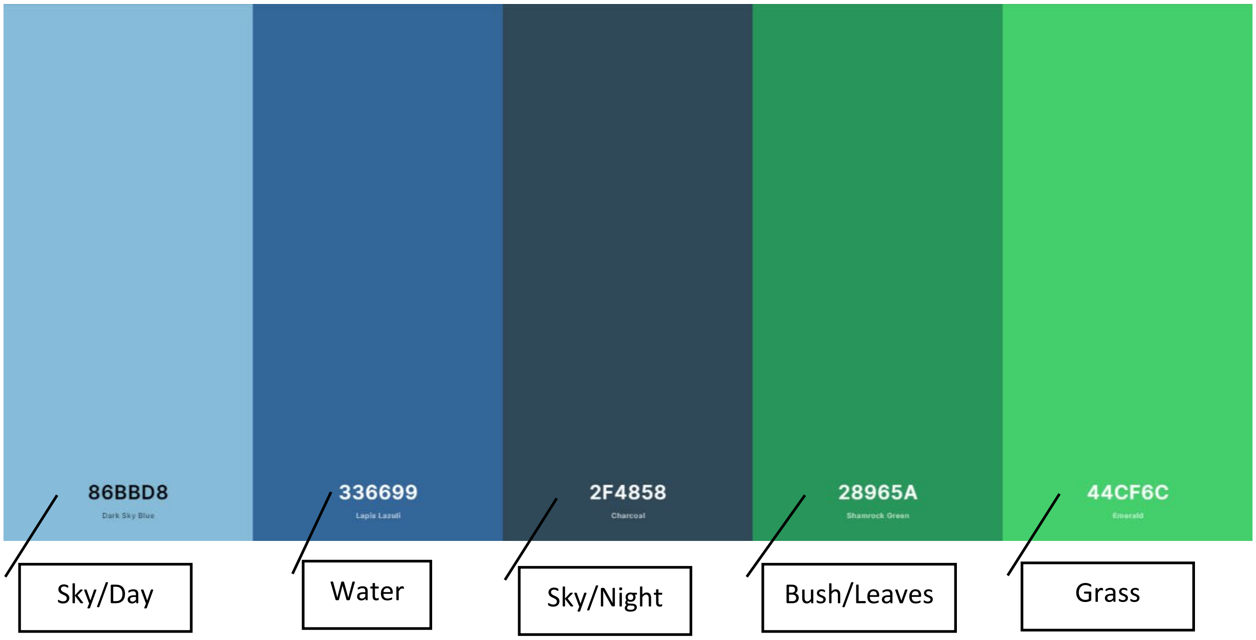
Final World 2 Level: Multiple Portals used with loops. Maybe too hard?

COLOUR PALETTES

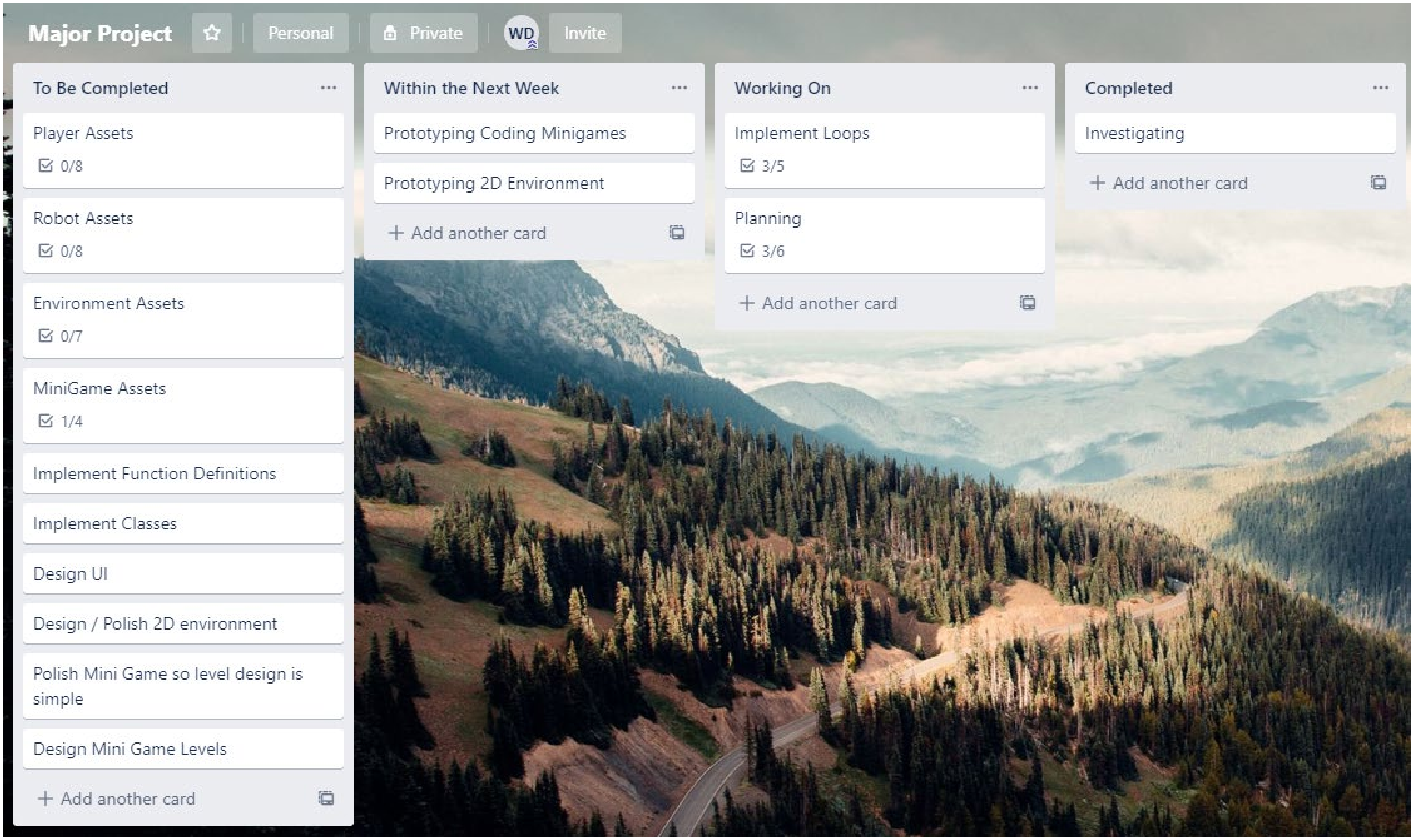
Block Colour Theme



World Design



PROJECT MANAGEMENT ON TRELLO



To Be Completed

- Player Assets ☑ 0/8
- Robot Assets ☑ 0/8
- Environment Assets ☑ 0/7
- MiniGame Assets ☑ 1/4
- Implement Function Definitions
- Implement Classes
- Design UI
- Design / Polish 2D environment
- Polish Mini Game so level design is simple
- Design Mini Game Levels

Within the Next Week

- Prototyping Coding Minigames
- Prototyping 2D Environment

Working On

- Implement Loops ☑ 3/5
- Planning ☑ 3/6

Completed

- Investigating

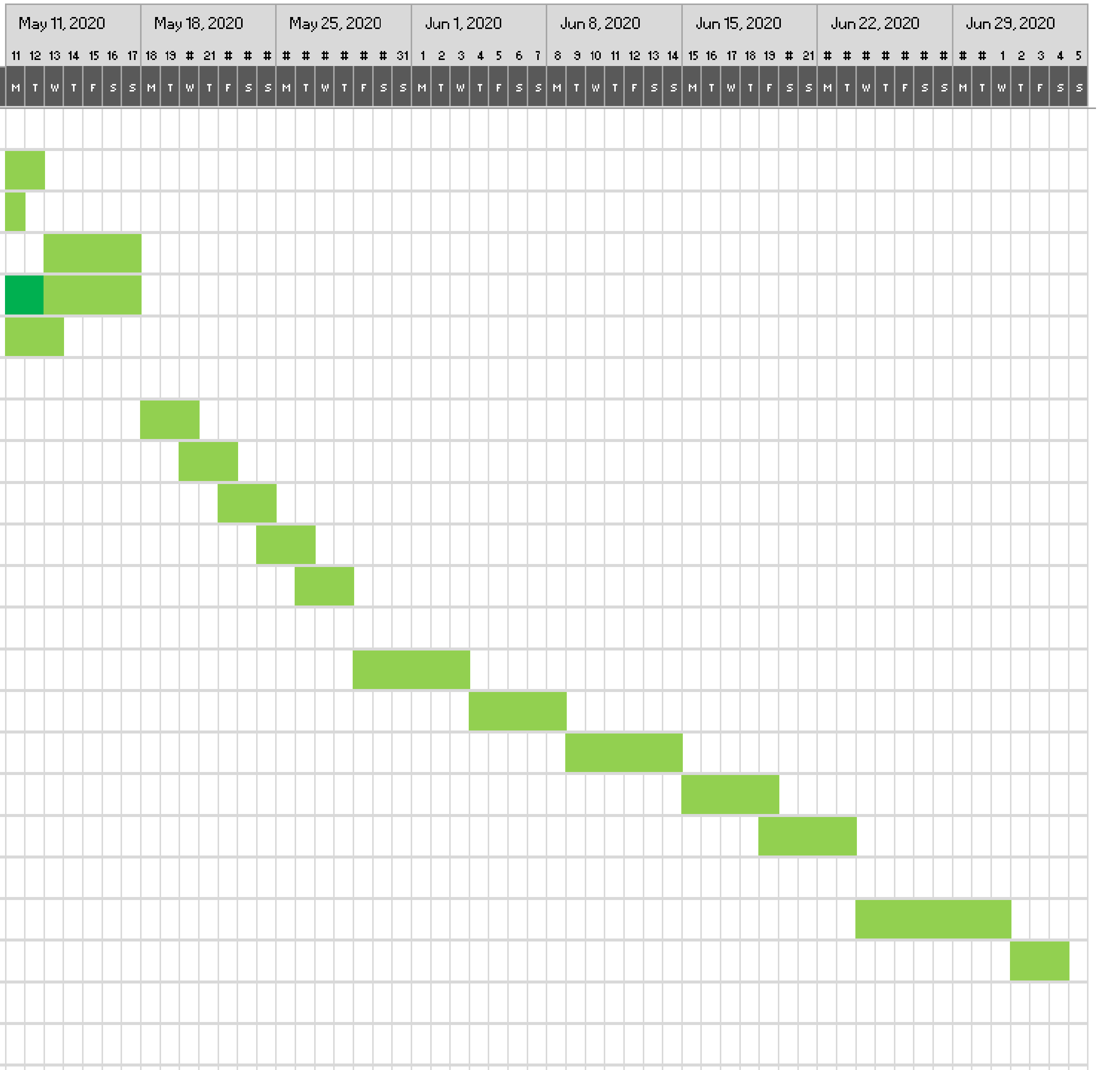
GAANT CHART

Will Davis

Project Start: 10/05/2020

Display Week: 1

TASK	ASSIGNED TO	PROGRESS	START	END
Planning				
Sketching	Will	20%	10-May-20	12-May-20
Create Gaant Chart	Will	10%	10-May-20	11-May-20
Digital Mockups	Will	0%	13-May-20	17-May-20
Code Prototyping	Will	40%	10-May-20	17-May-20
Sound Designing	Will	0%	11-May-20	13-May-20
Asset Creation				
Robot Sprite Sheet	Will	0%	18-May-20	20-May-20
Player Sprite Sheet	Will	0%	20-May-20	22-May-20
Environment Assets	Will	0%	22-May-20	24-May-20
Collectable Assets	Will	0%	24-May-20	26-May-20
Minigame Assets	Will	0%	26-May-20	28-May-20
Engine Development				
2D Engine	Will	0%	29-May-20	3-Jun-20
MiniGame Engine	Will	0%	4-Jun-20	8-Jun-20
Implement Loops	Will	0%	9-Jun-20	14-Jun-20
Implement Functions	Will	0%	15-Jun-20	19-Jun-20
Implement Classes	Will	0%	19-Jun-20	23-Jun-20
Beta Testing				
Designing Levels	Will	0%	24-Jun-20	1-Jul-20
Designing 2D World	Will	0%	2-Jul-20	9-Jul-20
Bug Fixing	Will	0%	10-Jul-20	15-Oct-20
PROJECT LAUNCH	Will		16-Oct-20	17-Oct-20



CODE TESTING

Attached File: Code Testing Video entitled “Code Testing.Mov”

Try Hyperlink → ▶

https://www.youtube.com/watch?v=9vsAE7fVbEE&feature=youtu.be&ab_channel=WillDavis

SOUND DESIGN

Sound Effects and voice overs will be recorded using the AKG-414 Condenser

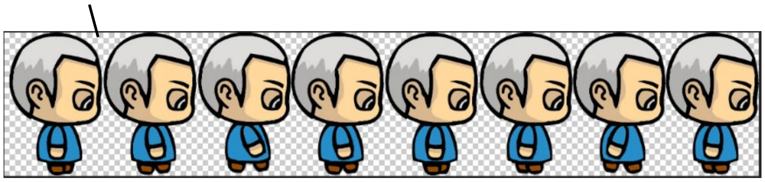
Backing tracks will be recorded with the Zoom-4HN Digital Recorder.

Medium	File Type	Properties
Recording	**WAV**	Sampling rate: 44.1kHz Bit Depth: 16 Channels: Stereo Polarity: Cardioid
Exporting Sound Effects	**OGG**	VBR: 0% Sampling Rate: 32 kHz Bit Depth: 16 Channels: Mono

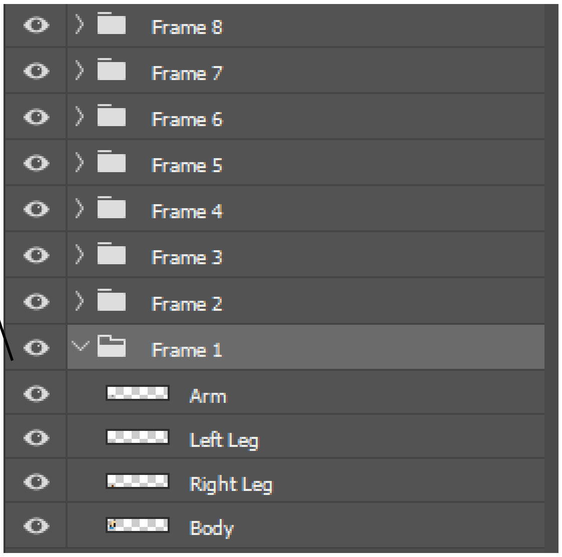
Sound Effect Needed	How will it be created - Foley
Click for clicking blocks into place	Sound of a highlighter lid clicking back on.
Walking sound	Walking over objects
Robot sounds	Drill

Graphic Design

This allows me to rotate the arms and legs, making the walking animation. I drew these using photoshop.

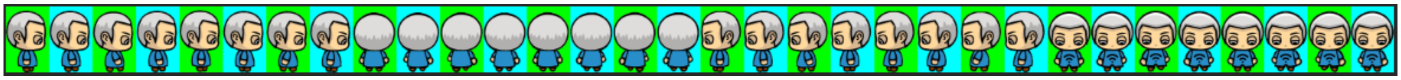


Each “Frame” given its own folder, then the player is split into 4 crucial parts, which can be manipulated freely of each other

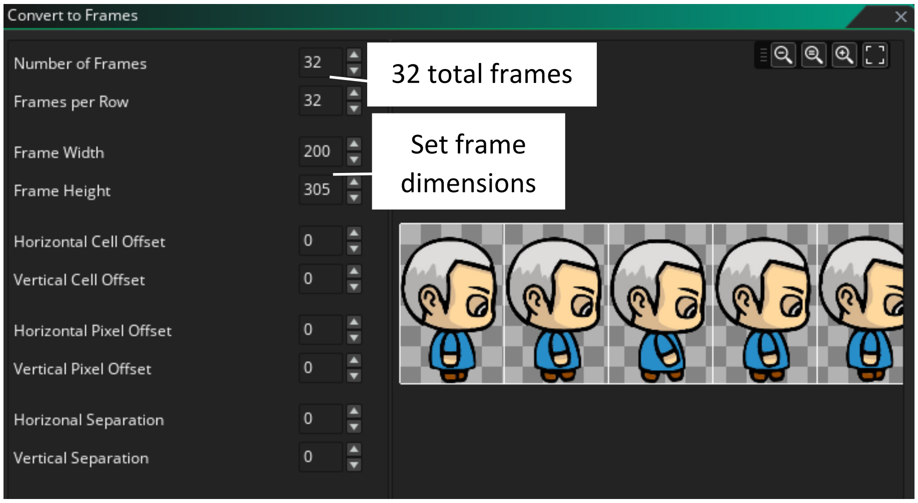


Frame 8
Frame 7
Frame 6
Frame 5
Frame 4
Frame 3
Frame 2
Frame 1
Arm
Left Leg
Right Leg
Body

Then this was repeated for each direction and put into a new Photoshop folder. 32 equisized rectangles were used to equally space each frame. This was exported as a PNG without the rectangles.



GameMakers Import_Strip_Image:

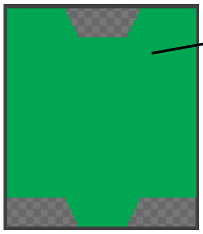


Convert to Frames

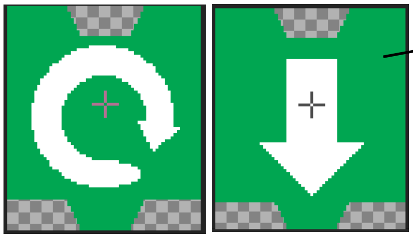
Number of Frames 32
Frames per Row 32
Frame Width 200
Frame Height 305
Horizontal Cell Offset 0
Vertical Cell Offset 0
Horizontal Pixel Offset 0
Vertical Pixel Offset 0
Horizonal Separation 0
Vertical Separation 0

32 total frames

Set frame dimensions



Tesseractable Duplicatable block for making new sprites within GameMaker

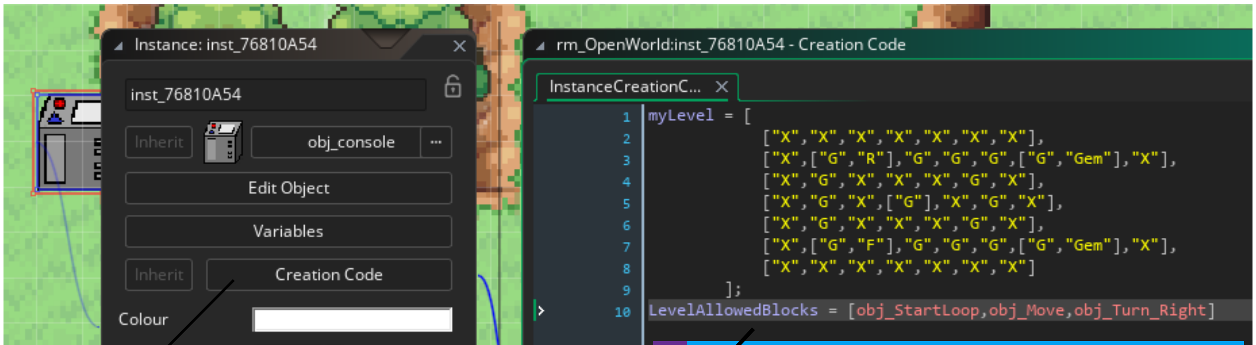


Draw inscriptions on top

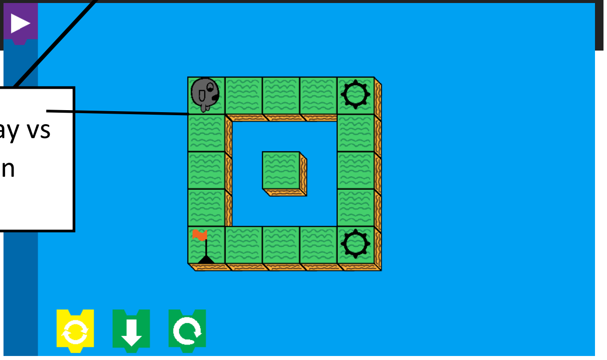
Level Design Code Engine

In order to make designing levels as easy as possible, an engine was created so that levels could be stored in a multidimensional array, then rendered to the screen, no matter what is stored within the array. Also, the code "blocks" allowed to use can be specified for each level.

Thing	Shorthand
Wall/No Block	"X"
Ground	"G"
Gem	"Gem"
Robot	"R"
Flag	"F"
[Colour] Portal	"[First Letter of Colour]P", ["Position to Send to"]
Greyed out Block	"Y" (Only if in make your own level room)

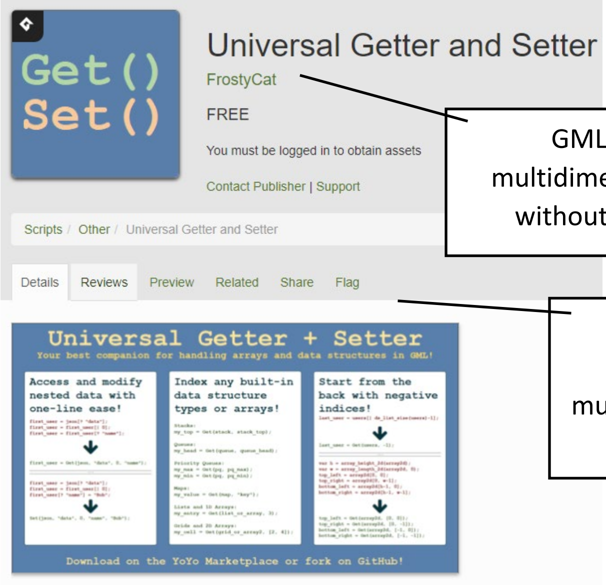


```
myLevel = [
            ["X","X","X","X","X","X","X"],
            ["X",["G","R"],"G","G","G",["G","Gem"],"X"],
            ["X","G","X","X","X","G","X"],
            ["X","G","X",["G"],"X","G","X"],
            ["X","G","X","X","X","G","X"],
            ["X",["G","F"],"G","G","G",["G","Gem"],"X"],
            ["X","X","X","X","X","X","X"]
        ];
LevelAllowedBlocks = [obj_StartLoop,obj_Move,obj_Turn_Right]
```



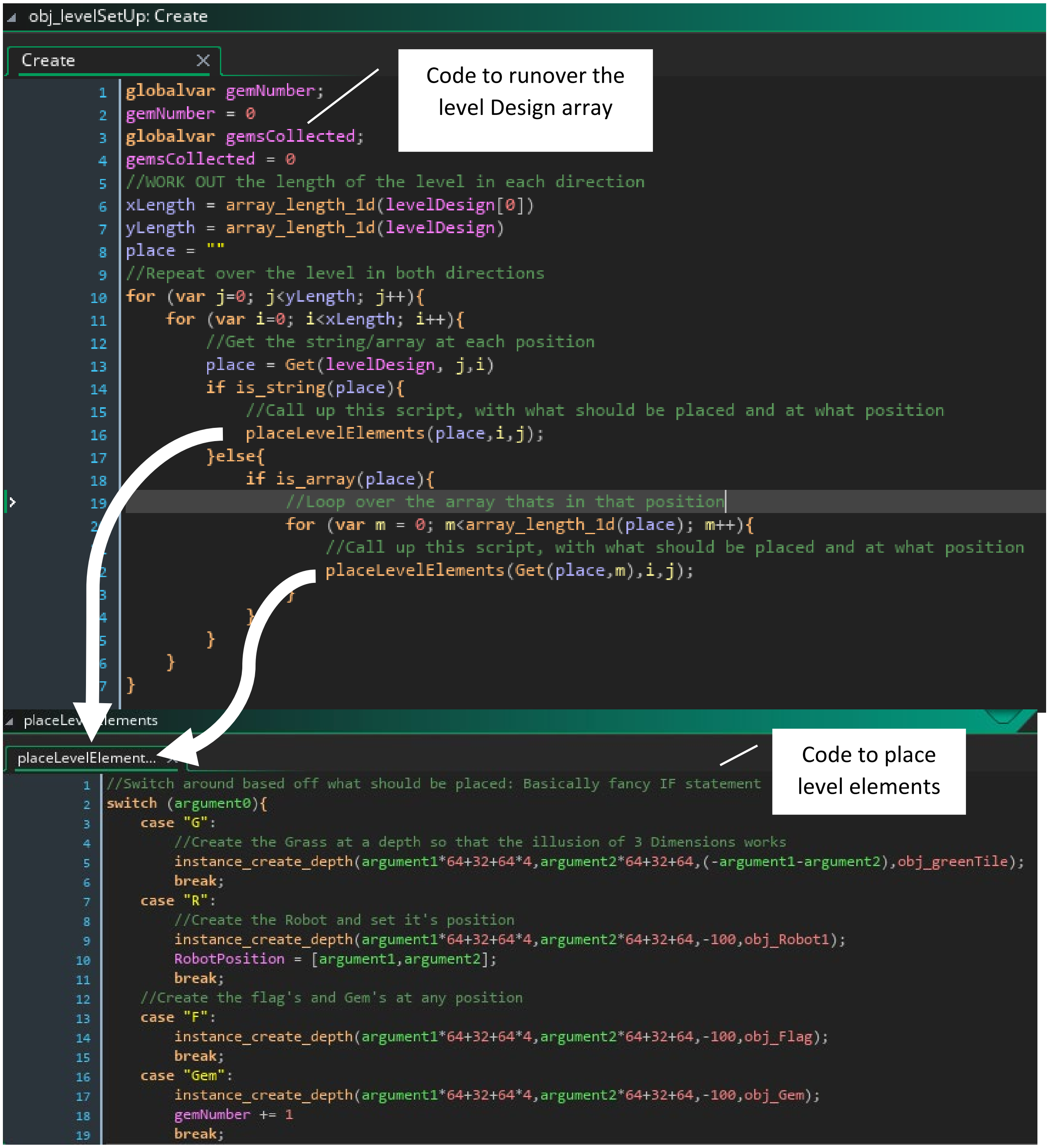
Level_Design and Allowed_Blocks stored in the instance code of the console.

Level design in array vs output on screen


Get()
Set()
Universal Getter and Setter
FrostyCat
FREE
You must be logged in to obtain assets
Contact Publisher | Support
Scripts / Other / Universal Getter and Setter
Details
Reviews
Preview
Related
Share
Flag
Universal Getter + Setter
Your best companion for handling arrays and data structures in GML!
Access and modify nested data with one-line ease!
Index any built-in data structure types or arrays!
Start from the back with negative indices!
Download on the YoYo Marketplace or fork on GitHub!

GML can't use multidimensional-arrays without this add-on.

Note: As of GML 2.3, support for multidimensional-arrays has been added.



Code to runover the level Design array

```
globalvar gemNumber;
gemNumber = 0
globalvar gemsCollected;
gemsCollected = 0
//WORK OUT the length of the level in each direction
xLength = array_length_1d(levelDesign[0])
yLength = array_length_1d(levelDesign)
place = ""
//Repeat over the level in both directions
for (var j=0; j<yLength; j++){
    for (var i=0; i<xLength; i++){
        //Get the string/array at each position
        place = Get(levelDesign, j,i)
        if is_string(place){
            //Call up this script, with what should be placed and at what position
            placeLevelElements(place,i,j);
        }else{
            if is_array(place){
                //Loop over the array thats in that position
                for (var m = 0; m<array_length_1d(place); m++){
                    //Call up this script, with what should be placed and at what position
                    placeLevelElements(Get(place,m),i,j);
                }
            }
        }
    }
}
```

Code to place level elements

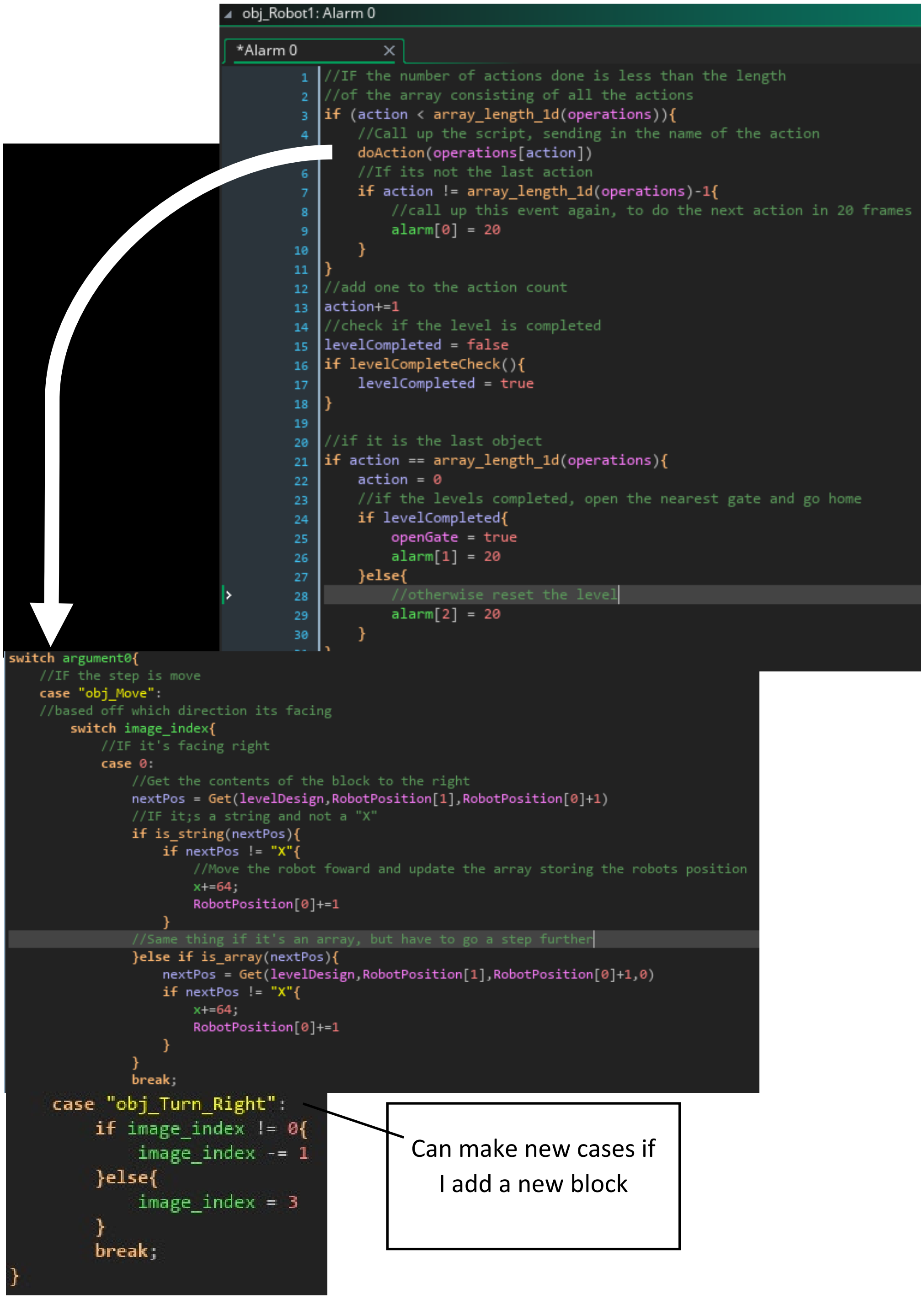
```
//Switch around based off what should be placed: Basically fancy IF statement
switch (argument0){
    case "G":
        //Create the Grass at a depth so that the illusion of 3 Dimensions works
        instance_create_depth(argument1*64+32+64*4,argument2*64+32+64,(-argument1-argument2),obj_greenTile);
        break;
    case "R":
        //Create the Robot and set it's position
        instance_create_depth(argument1*64+32+64*4,argument2*64+32+64,-100,obj_Robot1);
        RobotPosition = [argument1,argument2];
        break;
    //Create the flag's and Gem's at any position
    case "F":
        instance_create_depth(argument1*64+32+64*4,argument2*64+32+64,-100,obj_Flag);
        break;
    case "Gem":
        instance_create_depth(argument1*64+32+64*4,argument2*64+32+64,-100,obj_Gem);
        gemNumber += 1
        break;
```

Analysis

While writing this code I ensured it was scalable, meaning I could easily add new elements without altering code, this was helpful in the long run.

doAction Engine

This engine was created to move the robot within the minigame, so that it follows the commands.



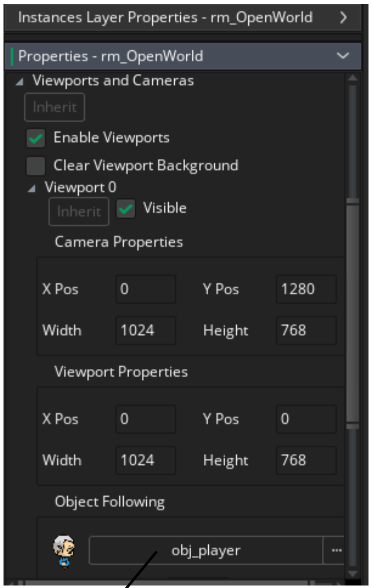
```
//IF the number of actions done is less than the length
//of the array consisting of all the actions
if (action < array_length_1d(operations)){
    //Call up the script, sending in the name of the action
    doAction(operations[action])
    //If its not the last action
    if action != array_length_1d(operations)-1{
        //call up this event again, to do the next action in 20 frames
        alarm[0] = 20
    }
}
//add one to the action count
action+=1
//check if the level is completed
levelCompleted = false
if levelCompleteCheck(){
    levelCompleted = true
}

//if it is the last object
if action == array_length_1d(operations){
    action = 0
    //if the levels completed, open the nearest gate and go home
    if levelCompleted{
        openGate = true
        alarm[1] = 20
    }else{
        //otherwise reset the level
        alarm[2] = 20
    }
```

```
switch argument0{
    //IF the step is move
    case "obj_Move":
    //based off which direction its facing
        switch image_index{
            //IF it's facing right
            case 0:
                //Get the contents of the block to the right
                nextPos = Get(levelDesign,RobotPosition[1],RobotPosition[0]+1)
                //IF it;s a string and not a "X"
                if is_string(nextPos){
                    if nextPos != "X"{
                        //Move the robot foward and update the array storing the robots position
                        x+=64;
                        RobotPosition[0]+=1
                    }
                //Same thing if it's an array, but have to go a step further
                }else if is_array(nextPos){
                    nextPos = Get(levelDesign,RobotPosition[1],RobotPosition[0]+1,0)
                    if nextPos != "X"{
                        x+=64;
                        RobotPosition[0]+=1
                    }
                }
                break;
```

```
    case "obj_Turn_Right":
        if image_index != 0{
            image_index -= 1
        }else{
            image_index = 3
        }
        break;
}
```

Can make new cases if I add a new block

Open World Level Design



Instances Layer Properties - rm_OpenWorld

Properties - rm_OpenWorld

Viewports and Cameras

Inherit

☑ Enable Viewports

□ Clear Viewport Background

Viewport 0

Inherit ☑ Visible

Camera Properties

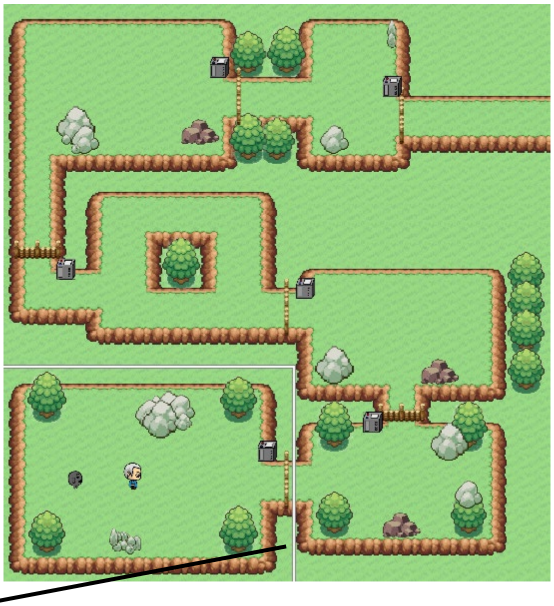
X Pos	0	Y Pos	1280
Width	1024	Height	768

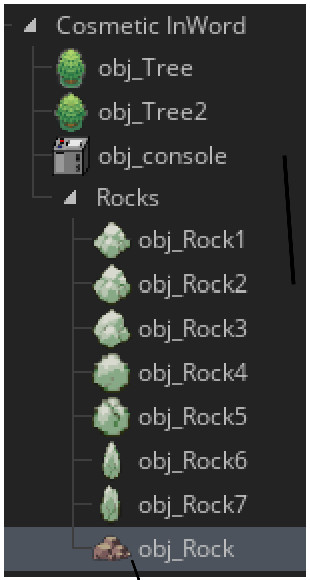
Viewport Properties

X Pos	0	Y Pos	0
Width	1024	Height	768

Object Following

obj_player



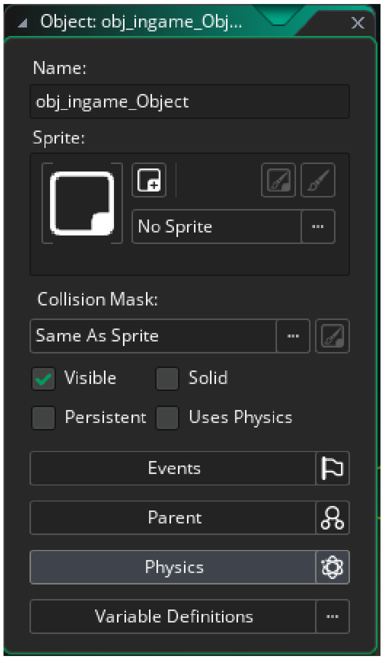


- Cosmetic InWord
  - obj_Tree
  - obj_Tree2
  - obj_console
  - Rocks
    - obj_Rock1
    - obj_Rock2
    - obj_Rock3
    - obj_Rock4
    - obj_Rock5
    - obj_Rock6
    - obj_Rock7
    - obj_Rock

Viewport follows player around

Aesthetic objects

Creates illusion of depth



Object: obj_ingame_Obj...

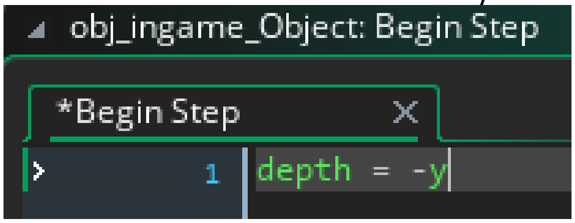
Name: obj_ingame_Object

Sprite: No Sprite

Collision Mask: Same As Sprite

☑ Visible □ Solid

□ Persistent □ Uses Physics

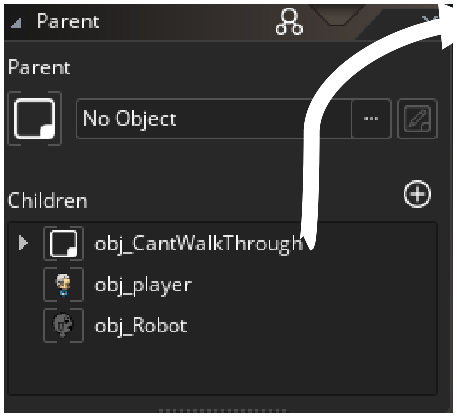


obj_ingame_Object: Begin Step

*Begin Step

```
depth = -y
```

Parent object for solid obstacles

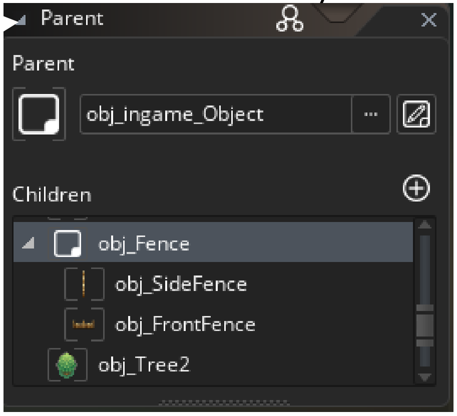


Parent

Parent: No Object

Children

- obj_CantWalkThrough
- obj_player
- obj_Robot



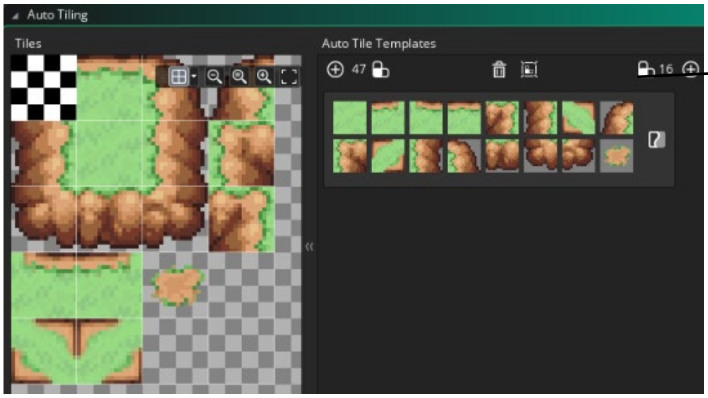
Parent

Parent: obj_ingame_Object

Children

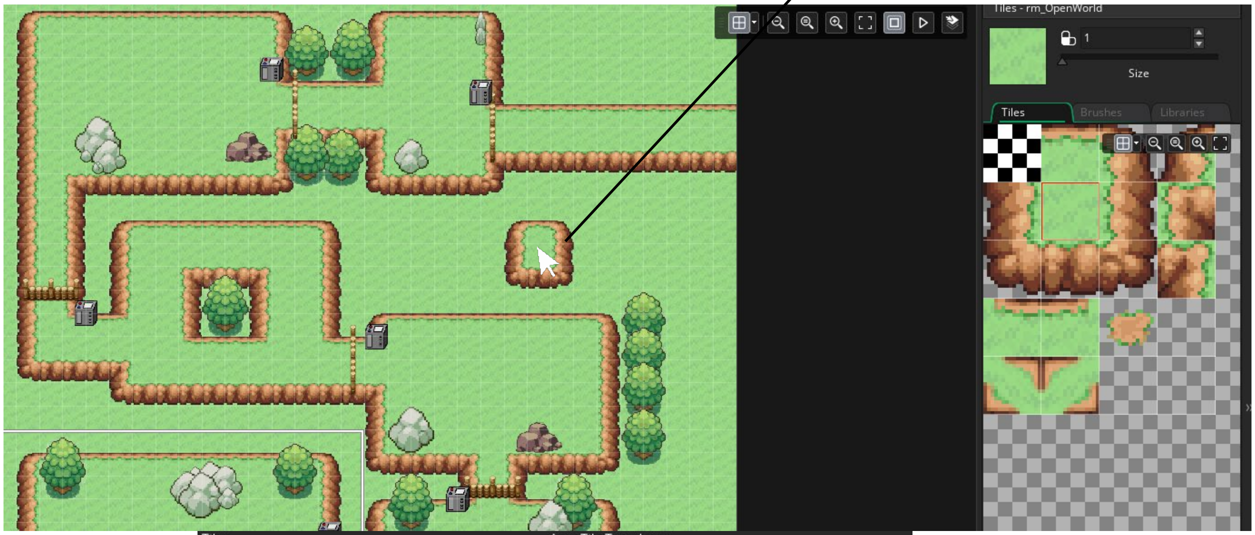
- obj_Fence
  - obj_SideFence
  - obj_FrontFence
- obj_Tree2

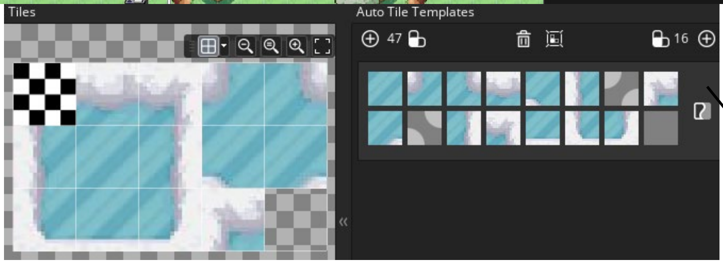
Auto-tiling System Setup



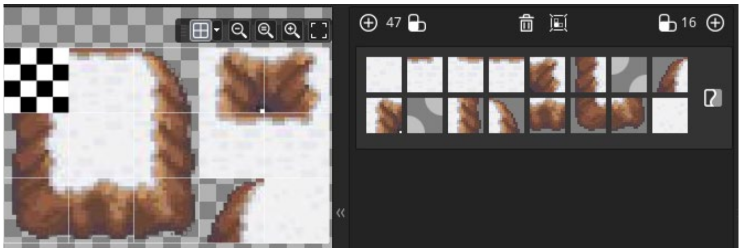
16 Tile set-up using GameMakers system

Drawing the middle tile automatically adds surrounding tiles in the correct order





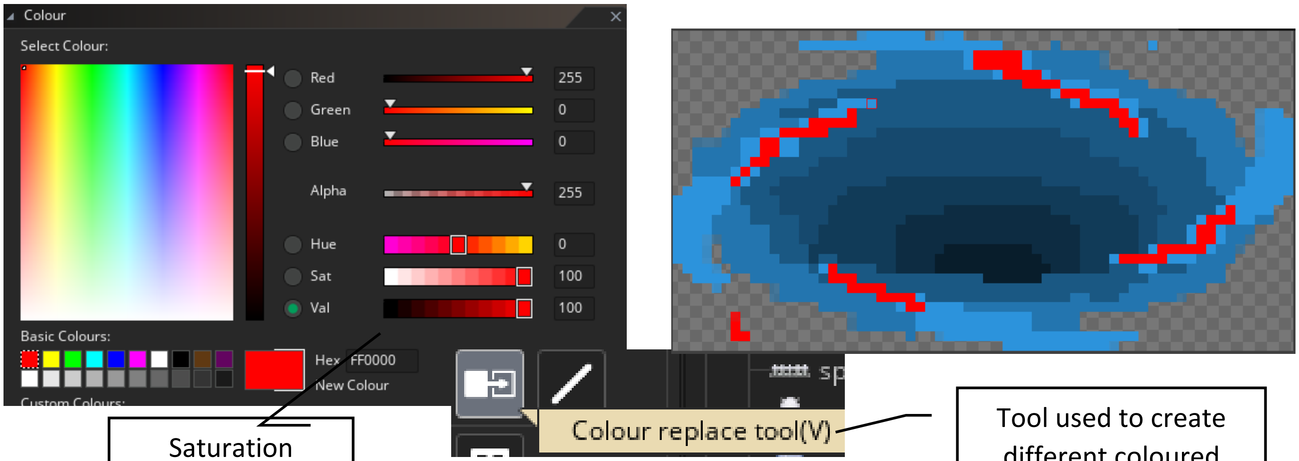
Ice and snow tiles created for snow



Analysis

This tiling system worked very well. A downside was auto-tiling couldn't be done using code, therefore this process could not be continuously generated.

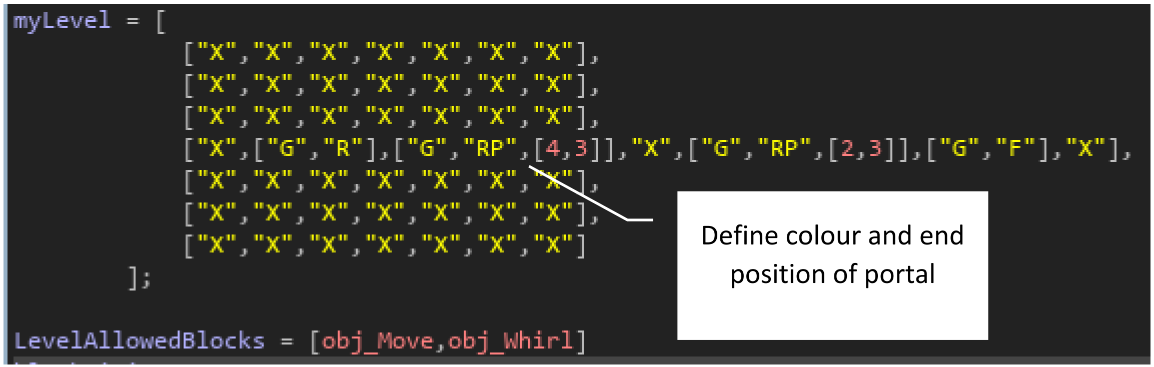
Portal System



Red 255
Green 0
Blue 0
Alpha 255
Hue 0
Sat 100
Val 100
Hex FF0000

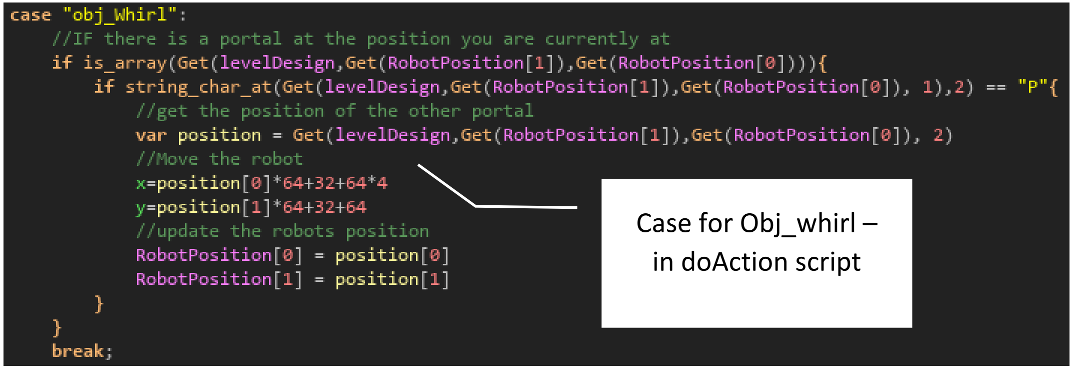
Saturation decreased for each layer.

Tool used to create different coloured portals



```
myLevel = [
            ["X","X","X","X","X","X","X"],
            ["X","X","X","X","X","X","X"],
            ["X","X","X","X","X","X","X"],
            ["X",["G","R"],["G","RP",[4,3]],"X",["G","RP",[2,3]],["G","F"],"X"],
            ["X","X","X","X","X","X","X"],
            ["X","X","X","X","X","X","X"],
            ["X","X","X","X","X","X","X"]
        ];

LevelAllowedBlocks = [obj_Move,obj_Whirl]
```

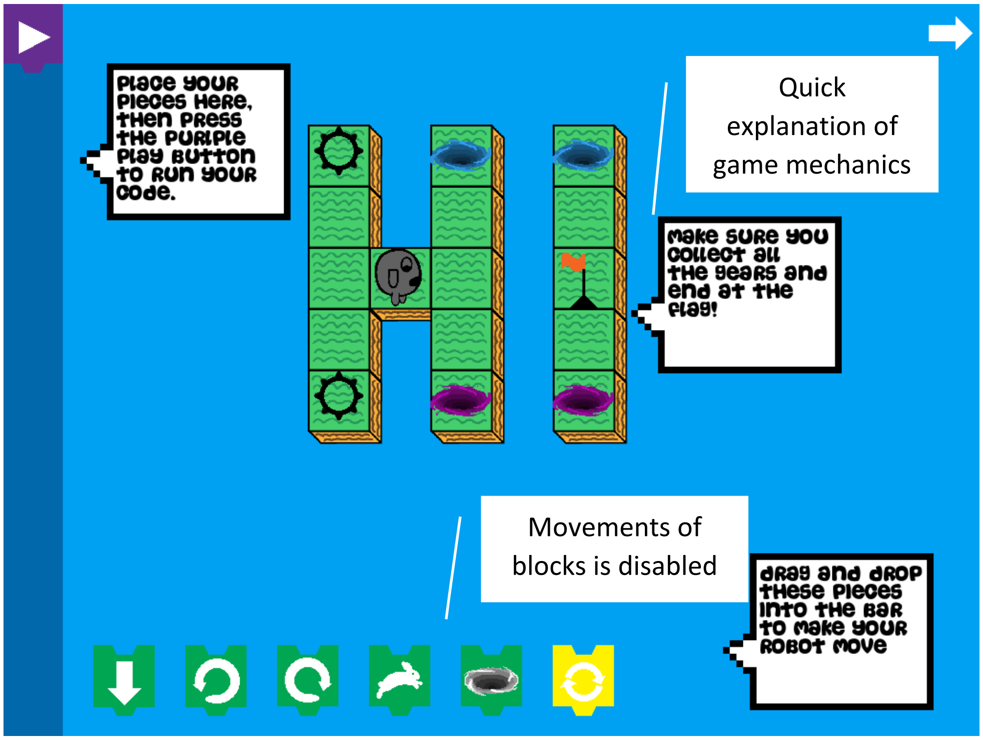
Define colour and end position of portal



```
case "obj_Whirl":
    //IF there is a portal at the position you are currently at
    if is_array(Get(levelDesign,Get(RobotPosition[1]),Get(RobotPosition[0]))){
        if string_char_at(Get(levelDesign,Get(RobotPosition[1]),Get(RobotPosition[0]), 1),2) == "P"{
            //get the position of the other portal
            var position = Get(levelDesign,Get(RobotPosition[1]),Get(RobotPosition[0]), 2)
            //Move the robot
            x=position[0]*64+32+64*4
            y=position[1]*64+32+64
            //update the robots position
            RobotPosition[0] = position[0]
            RobotPosition[1] = position[1]
        }
    }
    break;
```

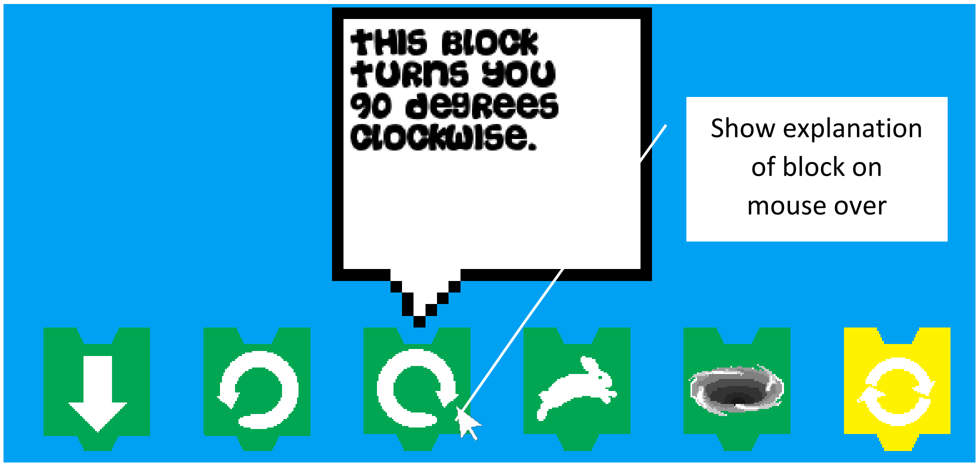
Case for Obj_whirl – in doAction script

Tutorial


Place your pieces here, then press the purple play button to run your code.
Make sure you collect all the gears and end at the flag!
Drag and drop these pieces into the bar to make your robot move

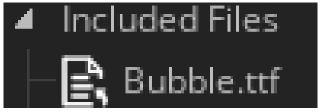
Quick explanation of game mechanics

Movements of blocks is disabled

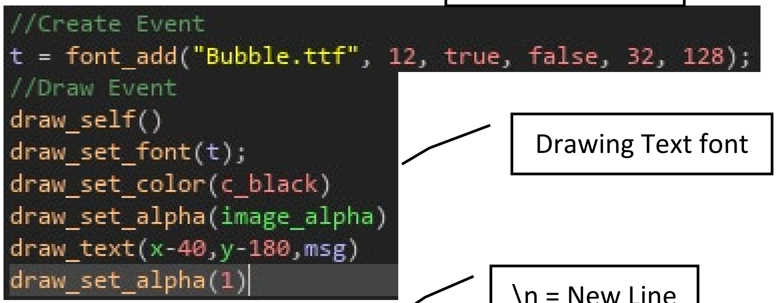


Show explanation of block on mouse over

Include font file



Defining font

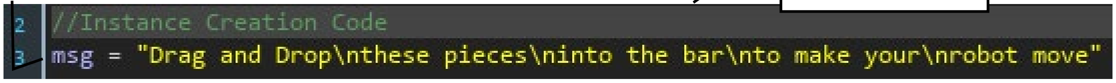


```
//Create Event
t = font_add("Bubble.ttf", 12, true, false, 32, 128);
//Draw Event
draw_self()
draw_set_font(t);
draw_set_color(c_black)
draw_set_alpha(image_alpha)
draw_text(x-40,y-180,msg)
draw_set_alpha(1)
```

Drawing Text font

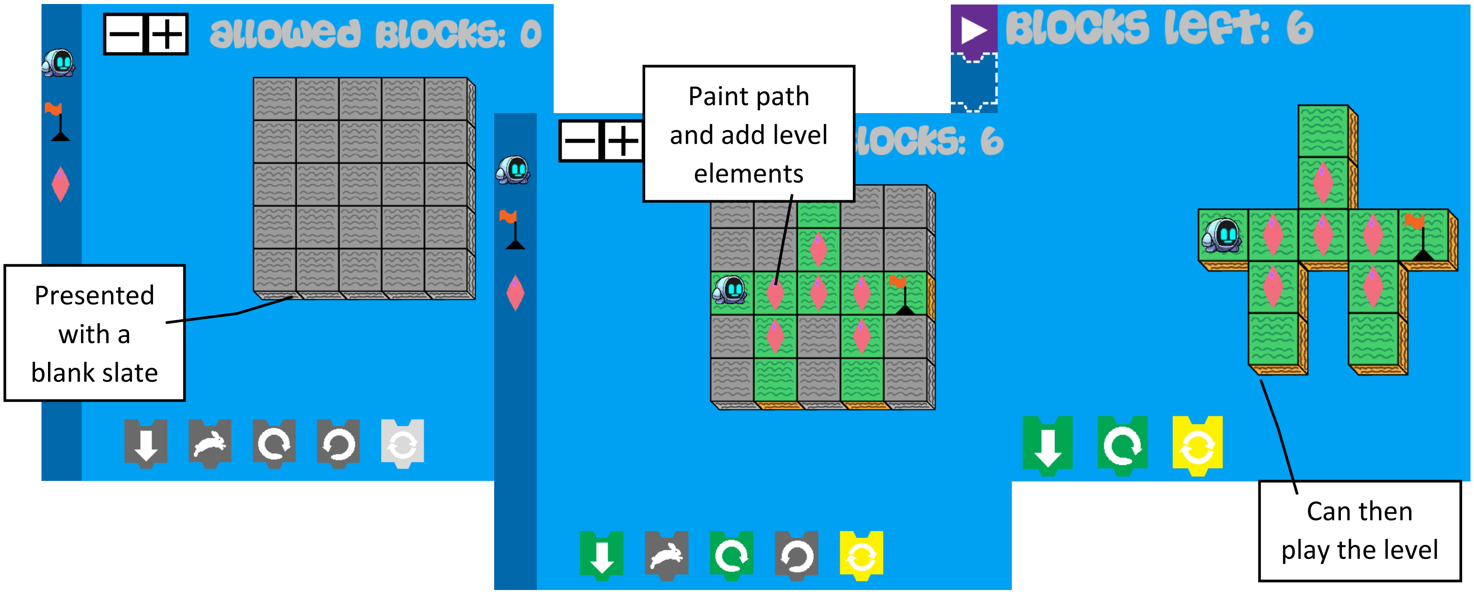
Define message

\n = New Line



```
//Instance Creation Code
msg = "Drag and Drop\nthese pieces\ninto the bar\nto make your\nrobot move"
```

Custom Level Maker System

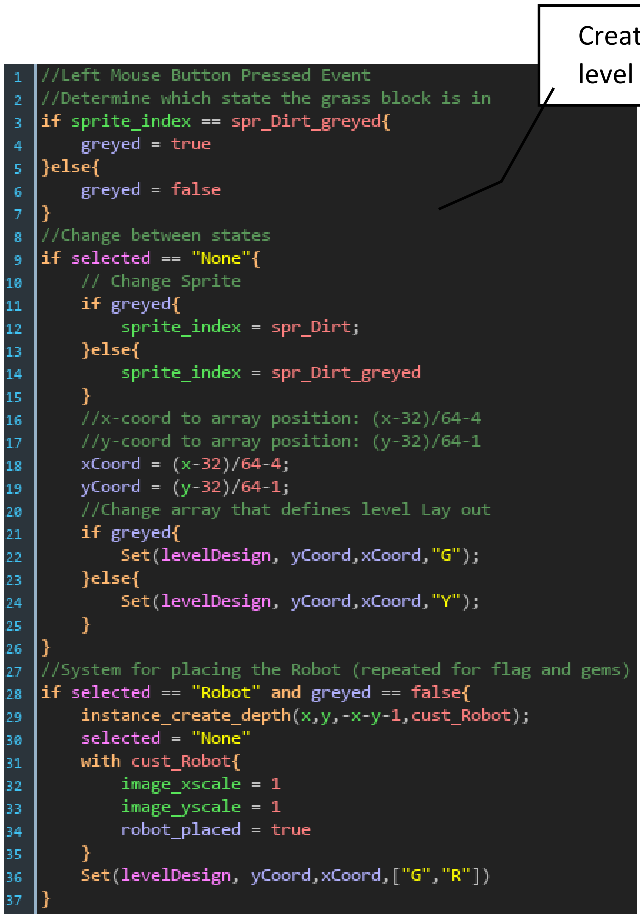


Presented with a blank slate

Paint path and add level elements

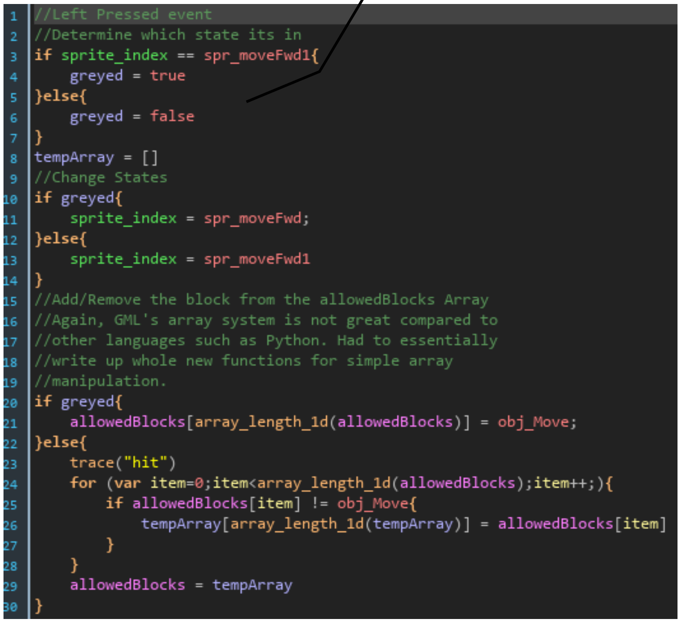
Can then play the level

Creating the level system



```
//Left Mouse Button Pressed Event
//Determine which state the grass block is in
if sprite_index == spr_Dirt_greyed{
    greyed = true
}else{
    greyed = false
}
//Change between states
if selected == "None"{
    // Change Sprite
    if greyed{
        sprite_index = spr_Dirt;
    }else{
        sprite_index = spr_Dirt_greyed
    }
    //x-coord to array position: (x-32)/64-4
    //y-coord to array position: (y-32)/64-1
    xCoord = (x-32)/64-4;
    yCoord = (y-32)/64-1;
    //Change array that defines level Lay out
    if greyed{
        Set(levelDesign, yCoord,xCoord,"G");
    }else{
        Set(levelDesign, yCoord,xCoord,"Y");
    }
}
//System for placing the Robot (repeated for flag and gems)
if selected == "Robot" and greyed == false{
    instance_create_depth(x,y,-x-y-1,cust_Robot);
    selected = "None"
    with cust_Robot{
        image_xscale = 1
        image_yscale = 1
        robot_placed = true
    }
    Set(levelDesign, yCoord,xCoord,["G","R"])
}
```

Selecting allowed Blocks system



```
//Left Pressed event
//Determine which state its in
if sprite_index == spr_moveFwd1{
    greyed = true
}else{
    greyed = false
}
tempArray = []
//Change States
if greyed{
    sprite_index = spr_moveFwd;
}else{
    sprite_index = spr_moveFwd1
}
//Add/Remove the block from the allowedBlocks Array
//Again, GML's array system is not great compared to
//other languages such as Python. Had to essentially
//write up whole new functions for simple array
//manipulation.
if greyed{
    allowedBlocks[array_length_1d(allowedBlocks)] = obj_Move;
}else{
    trace("hit")
    for (var item=0;item<array_length_1d(allowedBlocks);item++;){
        if allowedBlocks[item] != obj_Move{
            tempArray[array_length_1d(tempArray)] = allowedBlocks[item]
        }
    }
    allowedBlocks = tempArray
}
```

Evaluation

Effectiveness of requirements of design brief:

Outcome	Implemented	Modified	Not Implemented
Criterion 1:			
Character and Robot	✓		
Interactable Chests			✓
Gates that open after solving problems	✓		
Criterion 2:			
Moveable code blocks controlling robot	✓		
Simple “For Loop” blocks	✓		
User defined Functions			✓
Bug Fixing			✓
Classes			✓
Criterion 3			
Settings Screen	✓		
Minigame End Screen	✓		
Main Start Screen	✓		
Main End Screen	✓		
Criterion 4			
Multiple kingdoms for each coding element		✓	
5 levels per kingdom	✓		
Criterion 5			
Ambient Background noise	✓		
Sound Effects	✓		
Voice Overs			✓
Additional Criterion 1			
Custom Level maker	✓		

Summary:

Most criteria set out were successfully implemented. The omission of User functions, bug fixing, and classes led to the modification of the kingdom criteria. The first kingdom focused on loops. In order to differentiate the second kingdom, it was set in a snowy biome focusing on portals, a “gimmick” which loops, and other game elements can be explored with. Interactable chests were omitted from the final product, as it drew attention away from the focus, the educational minigames.

Effectiveness of design processes

Investigation and Planning

Originally, coming up with a concept was difficult, however, mindmaps assisted in the creative allowing an idea to be developed, which then was further expanded upon. Analysing similar products assisted with the development of levels, and coding concepts used throughout the minigames. This provided a solid set of ideas that were then developed and modified using an iterative agile development process.

Time Management

GAANT Chart and Kanban Boards

The GAANT chart was useful at the beginning for making sure there was enough time to complete all the features set out, however, the agile nature of the development made it difficult to adhere to. Following my Trello board with flexible elements updated weekly worked well to ensure that all the assets and coding features were produced in a timely and organised manner.

To Be Completed

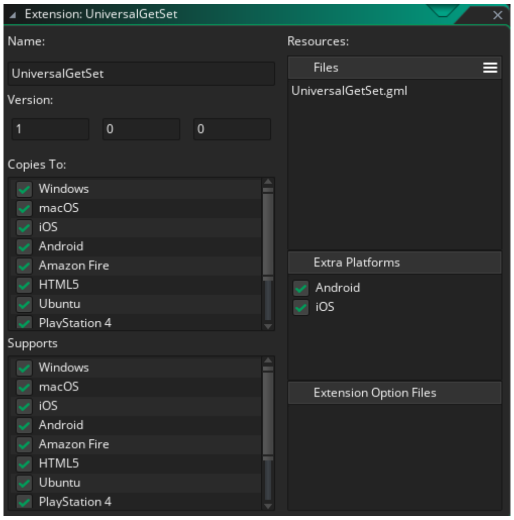
Completed Trello Board

Initial Testing of Code

Before beginning constructing the product, a very basic prototype of the minigames was made, to ensure that what I wanted to create was feasible. This was a crucial process to avoid loss of time by investing in a feature that was then not possible or too difficult to code

Working within limitations and constraints

At the beginning of construction, working with multidimensional arrays was extremely frustrating. I had to access online forums for assistance in finding a suitable solution. Scripts (functions) were imported that make using multidimensional arrays possible and simple.


Extension: UniversalGetSet

Name: UniversalGetSet

Version: 1 0 0

Copies To:
- ☑ Windows
- ☑ macOS
- ☑ iOS
- ☑ Android
- ☑ Amazon Fire
- ☑ HTML5
- ☑ Ubuntu
- ☑ PlayStation 4

Supports
- ☑ Windows
- ☑ macOS
- ☑ iOS
- ☑ Android
- ☑ Amazon Fire
- ☑ HTML5
- ☑ Ubuntu
- ☑ PlayStation 4

Resources:

Files

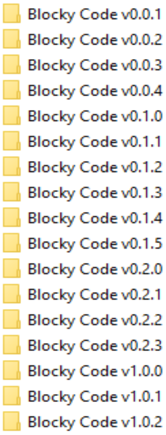
UniversalGetSet.gml

Extra Platforms
- ☑ Android
- ☑ iOS

Extension Option Files

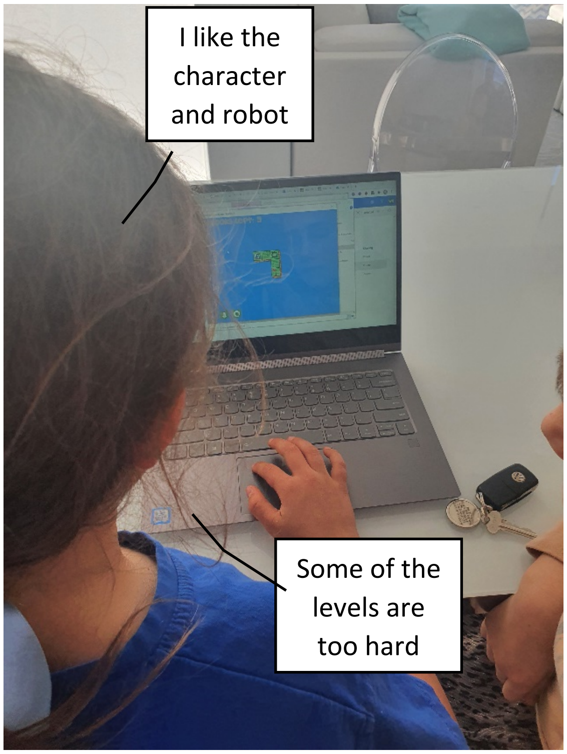
Other Processes that also worked well

Iterative File Saving allowed for backups of features designed, as sometimes bugs occur and transferring between home and school computers caused files to corrupt, this allowed me to have versions to fall back on without losing much development time.


Blocky Code v0.0.1
Blocky Code v0.0.2
Blocky Code v0.0.3
Blocky Code v0.0.4
Blocky Code v0.1.0
Blocky Code v0.1.1
Blocky Code v0.1.2
Blocky Code v0.1.3
Blocky Code v0.1.4
Blocky Code v0.1.5
Blocky Code v0.2.0
Blocky Code v0.2.1
Blocky Code v0.2.2
Blocky Code v0.2.3
Blocky Code v1.0.0
Blocky Code v1.0.1
Blocky Code v1.0.2

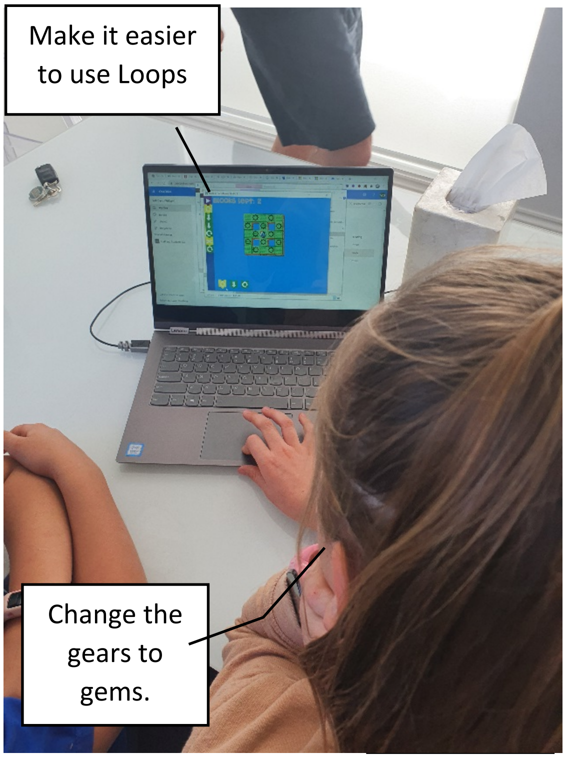
Testing with Target Audience

User Testing was done with two members of the target audience who gave me valuable feedback for suitable modifications:



I like the character and robot

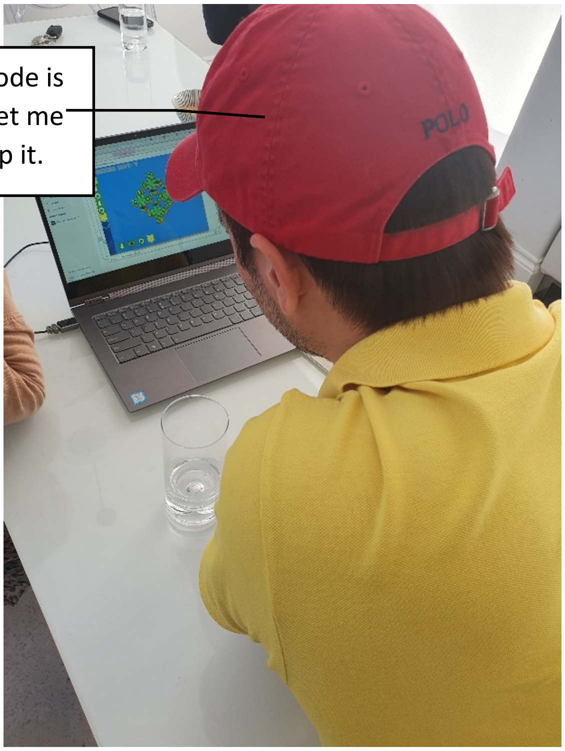
Some of the levels are too hard



Make it easier to use Loops

Change the gears to gems.

Their dad completed some levels. He had experience in programming so provided me with some helpful positive feedback.



While the code is executing, let me pause/stop it.

Key Takeaways:

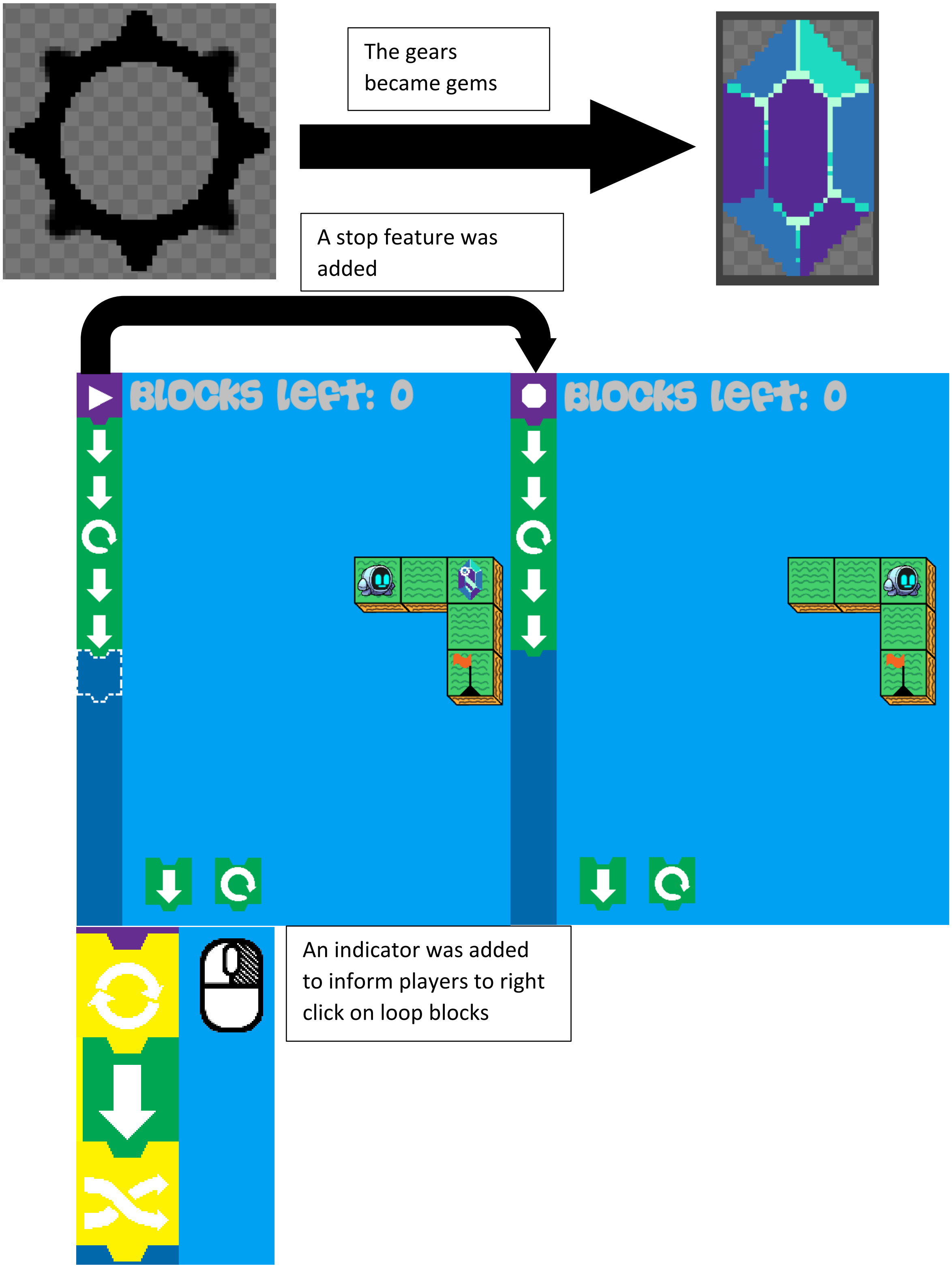
Stop feature for mid minigame.

Decrease difficulty of some levels/add hints

Explain loops better

Modifications

Multiple modifications were introduced due to the testing:



The gears became gems

A stop feature was added

An indicator was added to inform players to right click on loop blocks

Improvements / Recommendations

Accessibility

Accessibility features could be added for people who are colour blind or have learning disabilities. This could include altering the colour scheme, altering levels and accounting for other accessibility needs.

More Code Blocks / Features

Ideas for addition features within the minigames included walls you could climb, buttons and pushable boxes. These would add levels of intricacy to the game, allowing for more puzzles to be created. Introducing these could allow for new code blocks for actions such as “Push”, “Climb” or “Activate” again adding more features too the game.

Improved Level Design Features

An online system could be introduced in which levels could be uploaded and downloaded, this would be helpful with teachers designing levels for their students and people creating challenging levels for the general public or more experienced coders.

Investigate Publishing on Apple/Android

Through conversations with teachers in the junior-school, Apple-iPads are the primary technology used for teaching, so investigating porting the product to IOS/Android would be beneficial.

Extension of code learning (also recommended by junior-school teachers)

Variables, IF-Statements, and While-Loops are all features that could be added in future updates. Including these would require a major redesign of the drag and drop system but would vastly increase the coding knowledge the product would teach.

Conclusion

Accomplishing the goals set out at the beginning of this project, I am incredibly proud of the result, and believe with some additional features it could be released as a product. Ironically, I vastly improved my coding knowledge throughout the development, but importantly I got experience working on a large project and was able to improve my time management and other skills through that. I am excited to play through the finished product with my family and get the game into as many people’s hands as possible.

References:

Apple 2014, Playgrounds, version iPad, computer application, viewed 22 October 2020, <https://www.apple.com/au/swift/playgrounds/>.

Arthur 2017, Sci-fi-shwop-1, Mp3, OpenGameArt.org viewed 22 October 2020, <https://opengameart.org/content/sci-fi-shwop-1>.

Brunian 2017, BitGameSound.mp3, Mp3, OpenGameArt.org viewed 22 October 2020, <https://opengameart.org/content/simple-music>.

Butterscotch Shennanigens 2019, Levelhead, Computer Application, viewed 22 October 2020, <https://store.steampowered.com/app/792710/Levelhead/>.

Computer Scientist Job Future 2014, Graph, Geekwire, viewed 22 October 2020, <https://www.geekwire.com/2014/analysis-examining-computer-science-education-explosion/>.

Fences Autotile n.d., Image, DeviantArt, viewed 22 October 2020, <https://www.pinterest.com.au/pin/346988346263142661/?nic_v2=1a7CJcIx7>.

Freebies n.d., Image, MagicScarf, viewed 22 October 2020, <https://www.deviantart.com/magiscarf/art/Freebies-447772645>.

Funk, J 2018, How to Code a Sandcastle, N/A, N/A.

Google 2017, Celebrating 50 years of Kids Coding, viewed 22 October 2020, <https://www.google.com/doodles/celebrating-50-years-of-kids-coding>.

Job Outlook 2020, Software Engineers, Australian Government, viewed 22 October 2020, <https://joboutlook.gov.au/occupations/software-engineers?occupationCode=261313>.

Painting Cartoon n.d., Image, CleanPNG, viewed 22 October 2020, <https://www.cleanpng.com/png-pixel-art-sprite-rocks-734519/>.

Pixel Art Backgrounds - Tutorial: Skip n.d., Image, Behance, viewed 22 October 2020, <https://www.behance.net/gallery/65290819/Pixel-Art-Backgrounds-Tutorial-Skip>.

Pixel Gem n.d., Image, Dribble, viewed 22 October 2020, <https://dribbble.com/shots/5974208-Pixel-Gem>.

Portal gif 18 n.d., Image, gifImage.net, viewed 22 October 2020, <https://www.pinterest.com.au/pin/712905815991394283/?nic_v2=1a7CJcIx7>.

Spuispuin 2017, Won-orchestral-winning-jingle, Mp3, OpenGameArt.org viewed 22 October 2020, <https://opengameart.org/content/won-orchestral-winning-jingle>.

Tileset SnowyWorld n.d., Image, Deviantart, viewed 22 October 2020, <https://www.pinterest.com.au/pin/439804719856445117/?nic_v2=1a7CJcIx7>.

Vector pixel art sci fi airship n.d., Image, DreamsTime, viewed 22 October 2020, <https://www.dreamstime.com/vector-pixel-art-sci-fi-airship-scene-image188715500>.