If Memory Serves - Warwick New

What is "If Memory Serves"?

If Memory Serves is a game originally created in unity to attempt to teach younger audiences how pointers work. First developed by a team of researchers at Falmouth university.

The project is open source however was created using the unity game engine which is proprietary. So it became our teams task to port the game to an open source game engine.

Why Godot

We chose Godot initially because it looked like the most complete open source game engine we could find. However due to it's views on how objects should be placed makes levels unnecessarily difficult to generate levels on the fly.

Where Can I access this project and Godot?

Git repo: https://github.com/James120393/comp330_port

Godot: https://godotengine.org/

Sources:

[1] Monica M. McGill et al. 2017. If Memory Serves: Towards Designing and Evaluating a Game for Teaching Pointers to Undergraduate Students. In *Proceedings of the 2017 ITiCSE Working Group Reports* (ITiCSE '17). ACM, New York, NY, USA. DOI: https://doi.org/10.1145/3059009.3059037

[2] Chris Johnson et al. 2016. Game Development for Computer Science Education. In *Proceedings of the 2016 ITiCSE Working Group Reports* (ITiCSE '16). ACM, New York, NY, USA, 23-44. DOI: https://doi.org/10.1145/3024906.3024908

[3] Godot Engine Documentation, URL: http://docs.godotengine.org/en/stable/ [Online] Accessed - 08/12/2017

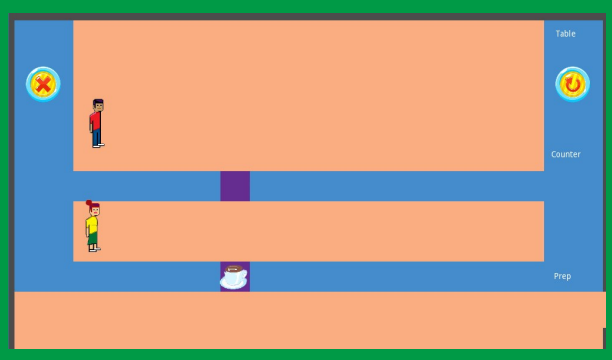
[4] Godot Engine, URL: https://godotengine.org/ [Online] Accessed - 08/12/2017, Paragraph One

My Contribution

In this project my goal was to allow the loading of levels in the Godot based port of If memory serves, on the same text files used in the original unity version. Pictured below is a copy of the text file I was attempting to decode.

```
; Assign a value to a variable in the stack.

; Level Settings
20 ; width
9 ; height
10 ; camera x
4.5 ; camera y
Val needs to move one cup of coffee from the preparation area t
A ; console target

; Initial Layout
--                  --
--                  --
--                  --
--                  --
-- &                --
--++++++x++++++++++--
--                  --
-- *                --
-------A------------
0 ; number of links

; Target Layout
--                  --
--                  --
--                  --
--                  --
--                  --
--++++++A++++++++++--
--                  --
--                  --
-------?------------
0 ; number of links

; Player Settings
addy valueTool ; addy's available tools
val valueTool ; val's available tools
par 2 ; maximum number of actions required for full reward

; Solution Code
begin
val = 'A';
a = val;
end
```

In the black box to the left is the initial layout of the first level in ascii form. Pictured below is the output from my algorithm which took this file as input. The main focus of the algorithm was to build the level aesthetically over functionally due to limitations imposed by the current version of the Godot engine and it's philosophy of how objects are handled in scenes.

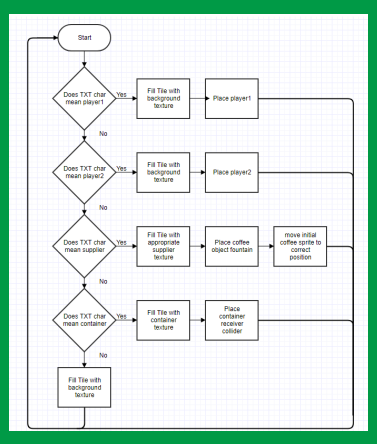


In the code to the Right is the initial function run when the tile map is first loaded.

It functions by using the new text file manipulation tools added to Godot in may and allow me to separate map data from value data

```
10
11 func _ready():
12     print (width)
13     for x in range(0, width):
14         for y in range(0, height):
15             set_cell(x,y,3)
16
17     #test new file reading
18     if file.file_exists("res://Game/Resources/Levels/level0-0.txt"):
19         file.open("res://Game/Resources/Levels/level0-0.txt", file.READ)
20         textMap = file.get_as_text()
21         file.close()
22         lineArray = textMap.split("\n")
23
24     # seperates aesthetic information
25     var isIntialMap = false
26     for i in lineArray:
27         if i == "; Initial Layout":
28             isIntialMap = true
29         if i.begins_with("-") and isIntialMap == true:
30             tileMapTextArray.append(i)
31         if i == "; Target Layout":
32             isIntialMap = false
33     print(tileMapTextArray)
```

To achieve this output I made use of Godot's tile map feature that with the advent of Godot 3.0 which only came out in march, can be easily manipulated to place in different tiles by a script. As shown in the script below.




```
# apply aesthetic layout to level
var position
var positionModifier
for y in range(0, tileMapTextArray.size()):
    for x in range(0, tileMapTextArray[y].length()):
        #determine which tile to place
        if tileMapTextArray[y][x] == "-":
            set_cell(x,y,1)
        elif tileMapTextArray[y][x] == "+":
            set_cell(x,y,1)

        # position players
        elif tileMapTextArray[y][x] == "*":
            position = map_to_world(Vector2(x,y))
            get_parent().get_node("Player2").set_global_position(position)
            set_cell(x,y,3)
        elif tileMapTextArray[y][x] == "&":
            position = map_to_world(Vector2(x,y))
            get_parent().get_node("Player1").set_global_position(position)
            set_cell(x,y,3)

        # place values
        elif tileMapTextArray[y][x] == "A":
            position = map_to_world(Vector2(x,y))
            positionModifier = Vector2(32,80)
            get_parent().get_node("Tween/Tween_Coffee").set_global_position(position + positionModifier)
            # pickup collider
            positionModifier = Vector2(32,0)
            get_parent().get_node("Bottom_Left").set_global_position(position + positionModifier)
            set_cell(x,y,2)

        elif tileMapTextArray[y][x] == "x":
            position = map_to_world(Vector2(x,y))
            positionModifier = Vector2(32,80*2)
            get_parent().get_node("Under_Left/CollisionShape2D").set_global_position(position + positionModifier)
            set_cell(x,y,8)

        # place background tiles in empty space
        else:
            set_cell(x,y,3)
```

You can also see that In the same loop as placing tiles I could place objects that were pre-placed in a scene in the correct location. However due to the behaviour of objects in the current version of Godot there is no object repository only scenes that inherit from one another. This makes it very difficult to create new objects that have behaviours that can influence other objects. This by extension makes generating levels dynamically outside of tile maps difficult and slow as each scripted object needs to created and placed into the scene. Therefore to test the script I created a scene with all the objects necessary for the first level in a blank scene ready to be repositioned based on the TXT version of the level shown here.