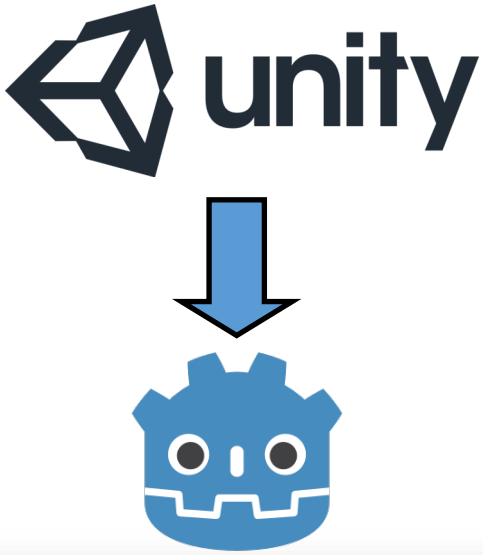
if memory serves

A game to “point” you in the right direction

If memory serves [1,2] is an open source game created by a group of researchers that attended the ACM Annual Conference of Innovation and Technology in Computer Science Education (ITiCSE 2017). The game revolves around teaching undergraduates pointers and was designed using a learning outcome based approach. This allowed for a game that facilitated educational games development within computer science.

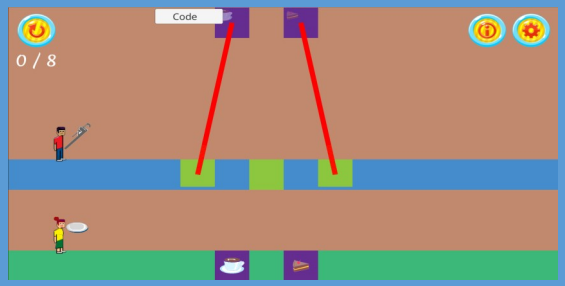
The project proposed in this poster is to port “If memory serves” to an open source games engine (we have chosen Godot) as it is currently being developed on Unity. The challenge here is learning a new games engine and scripting language along with it. Also, the art assets will need slight adjustments to better fit the new engine.



To port the game from Unity [5] to Godot [4], we made the decision to remake the game mechanics from scratch rather than translate each script one by one. This decision was made as it allowed us to build the game step by step making clear progress in each sprint rather than having little/nothing until the end when all the scrips were ported over.

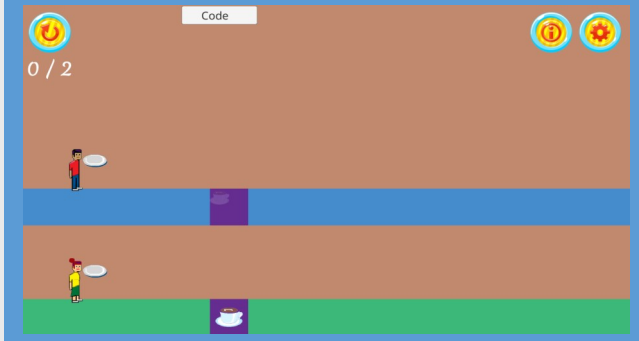
Plus this has allowed us to fix any issues as they come up, for example, the text parsing in Unity works differently to Godot. In Unity you can parse plain text files and it will work fine, but in Godot, the engine can only read one line at a time, this has lead to the use of JSON files in place of standard text.

This is the typical level in “if memory serves”, you have two characters Addy and Val, in each level, Val must pass Addy the correct item to pass onto the correct table, just like in a program where you must reference the correct object which is a pointer to the desired value. Each level has instructions on what needs doing as seen above. In the picture below is a level in which Addy and Val must create the correct pointers to allow the coffee and cake reach the correct tables. First Addy must create a pointer to the coffee table and Val must pick up the coffee and send it on, in this example only one pointer is allowed at a time so once the coffee has reached the table the pointer must be destroyed and a new one must be created for the cake table.



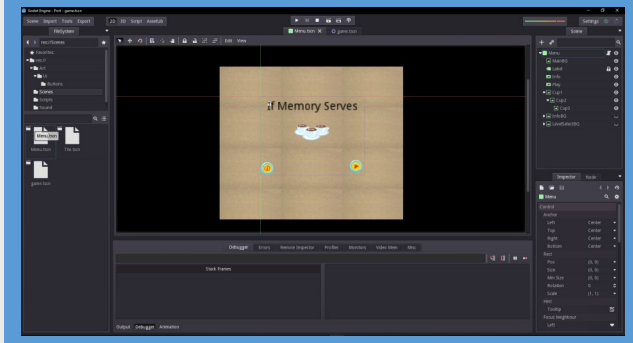
```
; Assign a value to a variable in the stack.

; Level Settings
20 ; width
9 ; height
10 ; camera x
4.5 ; camera y
Addy needs to move one cup of coffee from the preparation area to Val at the counter. ; instructions
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
```

This text file once parsed, translates into the image below. First setting up the dimensions of the level, then instructions and finally, taking in a series of symbols and replacing them with the appropriate in-game objects. Then the game will read what special setting the players will have, for example, Addy can either have a plate to carry coffee or a wrench to create/ change conveyor belts. Finally, the game will wait for the solution to be presented by the player and show the solution code to them.




```gdscript
extends Panel

# member variables here, example:

# var a=2
# var b="textvar"

func _ready():
    set_process_input(true)

func _input(event):
    if get_node("InfoBG/Cross").is_pressed():
        _on_cross_Button_pressed()
    if get_node("LevelSelectBG/Cross").is_pressed():
        _on_cross_Button_pressed()
    if get_node("Info").is_pressed():
        _on_info_Button_pressed()
    if get_node("Play").is_pressed():
        _on_play_Button_pressed()
    if get_node("LevelSelectBG/ScrollContainer/Control/HBoxContainer/Level1").is_pressed():
        _on_Level1_Buttone_pressed()

func _on_info_Button_pressed():
    get_node("InfoBG").set_hidden(false)
    get_node("Play").set_disabled(true)

func _on_cross_Button_pressed():
    get_node("InfoBG").set_hidden(true)
    get_node("LevelSelectBG").set_hidden(true)
    get_node("Play").set_disabled(false)

func _on_play_Button_pressed():
    get_node("LevelSelectBG").set_hidden(false)

func _on_Level1_Buttone_pressed():
    get_tree().change_scene("res://Scenes/game.tscn")
```

Godot uses its own scripting language called GDScript. GDScript is a high level, dynamically typed programming language used to create content. It uses a syntax similar to Python (blocks are indent-based and many keywords are similar) [3]. Above is a typical menu script that watches for button presses and calls the corresponding function to be executed.



The engine environment at first glance seems simple, though under this simple look is a powerful tool that was perfect for this project. Its 2D tools are elegant and requires little effort to create the environment you need. Though the downside to it is that the physics seem a little stiff at first as well as the 2D collision, it can be a little tricky to set up as you have to manually draw the collision box around the sprites. Another issue that has arisen is when creating tilesets to use in levels, the collision boxes were set up for trigger events but these do not translate to the game as intended and turn into blocking volumes when implemented into the level. This issue has not yet been resolved.

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

[5] Unity Engine, URL: https://unity3d.com/ [Online] Accessed - 08/12/2017