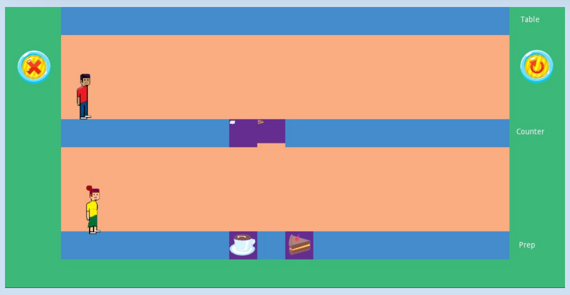
if memory serves

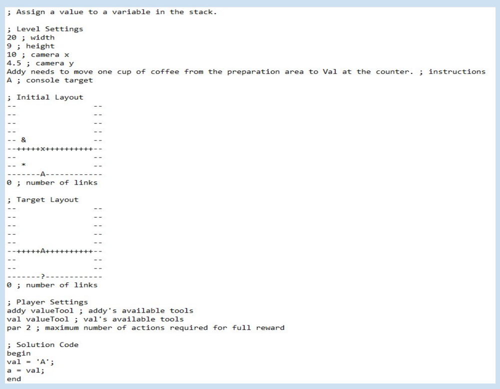
A game to "point" you in the right direction

James Hellman



The Art

The current art style remains the same as before, though these have been ported into GoDot.



```
; 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
```

Future Work

This project has provided the foundation to continue the porting task from Unity to GoDot.

There is still much that can be done in this port, for example, the remaining ready-made levels and the remaining learning objectives could be ported over. Plus implementing full automation for level loading to remove the need for manual construction, and implementing new art.

Open Source

This project is an open source project working with the Godot engine and source controlled by Github.

The project can be found here:

https://github.com/James120393/comp330_port

The engine is available for download here:

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

THE current state of the game includes Referencing and Pointers. There are many other learning objectives remaining to be implemented. In total there are currently 5 levels to complete, with a further 10 that could be implemented. The main game loop runs without issues, and currently, the player can complete the game.

My Contribution

For my contribution to the game I worked on:

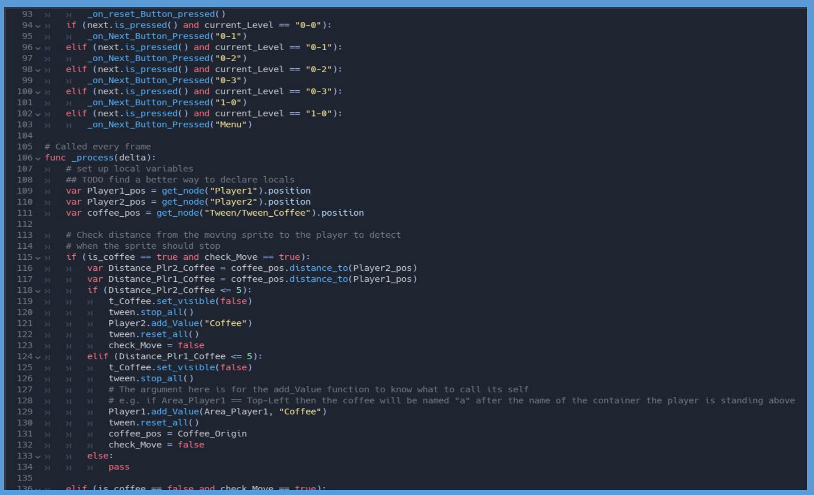
- The main gameplay loop
- Constructing of the levels
- Porting of art assets
- Game controls
- Level goals

Of the classes I created:

- Player 1&2.gd
- Container.gd
- Game.gd
- Level Selector.gd
- Menu.gd
- Text.gd
- Line.gd
- Area.gd

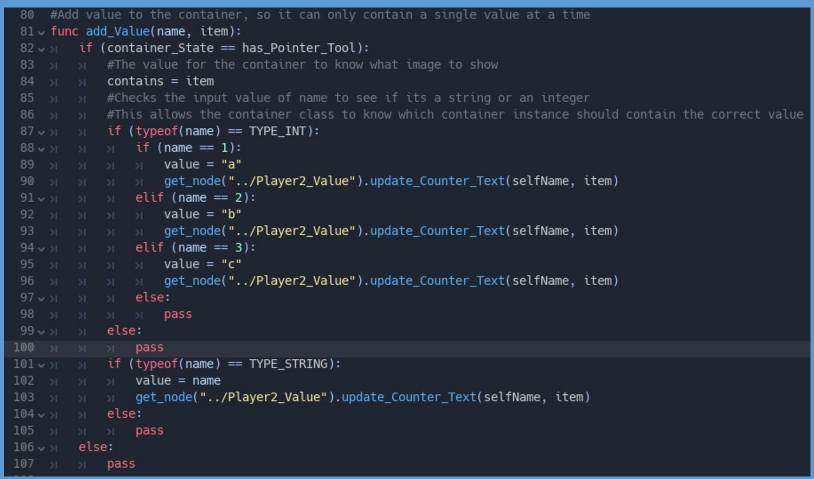
The code to change the value of the instances has been through several iterations, to begin with, there was a separate function for each value that the instance could hold. This lead to duplicate code and inefficiency when reading the values as the instance could hold more than one value. Which slowed down development time as each value had to be changed every time a value was input. This was modified so the instance could only accommodate one value at any given time, which prevented any bugs for miss referenced values. This change also helped development time as there was no need for several functions to be called.

The game class is the where the main game loop runs. This class communicates with the others classes giving them values. In the code imaged below, once the player pressed the pickup button the game will check the distance between the player and the object until the object has reached the player, at which point the value of the object will be passed into the relevant players class.

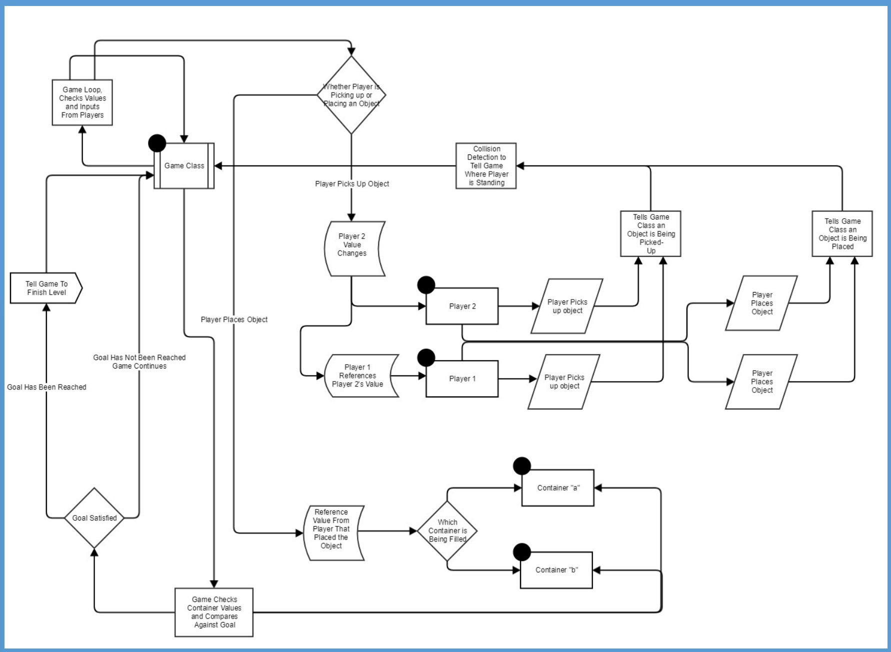


```
        _on_reset_Button_pressed()
    if (next.is_pressed() and current_Level == "0-0"):
        _on_Next_Button_Pressed("0-1")
    elif (next.is_pressed() and current_Level == "0-1"):
        _on_Next_Button_Pressed("0-2")
    elif (next.is_pressed() and current_Level == "0-2"):
        _on_Next_Button_Pressed("0-3")
    elif (next.is_pressed() and current_Level == "0-3"):
        _on_Next_Button_Pressed("1-0")
    elif (next.is_pressed() and current_Level == "1-0"):
        _on_Next_Button_Pressed("Menu")

# Called every frame
func _process(delta):
    # set up local variables
    ## TODO find a better way to declare locals
    var Player1_pos = get_node("Player1").position
    var Player2_pos = get_node("Player2").position
    var coffee_pos = get_node("Tween/Tween_Coffee").position

    # Check distance from the moving sprite to the player to detect
    # when the sprite should stop
    if (is_coffee == true and check_Move == true):
        var Distance_Plr2_Coffee = coffee_pos.distance_to(Player2_pos)
        var Distance_Plr1_Coffee = coffee_pos.distance_to(Player1_pos)
        if (Distance_Plr2_Coffee <= 5):
            t_Coffee.set_visible(false)
            tween.stop_all()
            Player2.add_Value("Coffee")
            tween.reset_all()
            check_Move = false
        elif (Distance_Plr1_Coffee <= 5):
            t_Coffee.set_visible(false)
            tween.stop_all()
            # The argument here is for the add_Value function to know what to call its self
            # e.g. if Area_Player1 == Top-Left then the coffee will be named "a" after the name of the container the player is standing above
            Player1.add_Value(Area_Player1, "Coffee")
            tween.reset_all()
            coffee_pos = Coffee_Origin
            check_Move = false
        else:
            pass

    elif (is_coffee == false and check_Move == true):
```

Other than learning a new language and a new engine, the most challenging aspect of this project was simulating the referencing and pointer system. The image to the below is a code exert that shows the function that the Player and Container class use to take in a value.



```
#Add value to the container, so it can only contain a single value at a time
func add_Value(name, item):
    if (container_State == has_Pointer_Tool):
        #The value for the container to know what image to show
        contains = item
        #Checks the input value of name to see if its a string or an integer
        #This allows the container class to know which container instance should contain the correct value
        if (typeof(name) == TYPE_INT):
            if (name == 1):
                value = "a"
                get_node("../Player2_Value").update_Counter_Text(selfName, item)
            elif (name == 2):
                value = "b"
                get_node("../Player2_Value").update_Counter_Text(selfName, item)
            elif (name == 3):
                value = "c"
                get_node("../Player2_Value").update_Counter_Text(selfName, item)
            else:
                pass
        else:
            pass
        if (typeof(name) == TYPE_STRING):
            value = name
            get_node("../Player2_Value").update_Counter_Text(selfName, item)
        else:
            pass
    else:
        pass
```


Game Loop, Checks Values and Inputs From Players
Game Class
Whether Player is Picking up or Placing an Object
Player Picks Up Object
Collision Detection to Tell Game Where Player is Standing
Player 2 Value Changes
Tells Game Class an Object is Being Picked-Up
Tells Game Class an Object is Being Placed
Tell Game To Finish Level
Player 2
Player Picks up object
Player Places Object
Player Places Object
Goal Has Not Been Reached Game Continues
Goal Has Been Reached
Player 1 References Player 2's Value
Player 1
Player Picks up object
Player Places Object
Container "a"
Goal Satisfied
Reference Value From Player That Placed the Object
Which Container is Being Filled
Container "b"
Game Checks Container Values and Compares Against Goal

This UML diagram describes the basic game loop. It shows the basic interaction between each class while the game is running.

Player 2 picks up an object, which then triggers a function within the game code that changes player 2's value to "A"

Player 2 then places an object in container "a" which triggers a function to change the value of the container to reference Player 2's value.

Player 1 picks up an object and now references Player 2, giving it the same value.

Player 1 places object and container "b" which now references Player 1's value making it the same as Player 2's Value which is "A"

The game then checks win condition and if win condition satisfied then the level finishes.