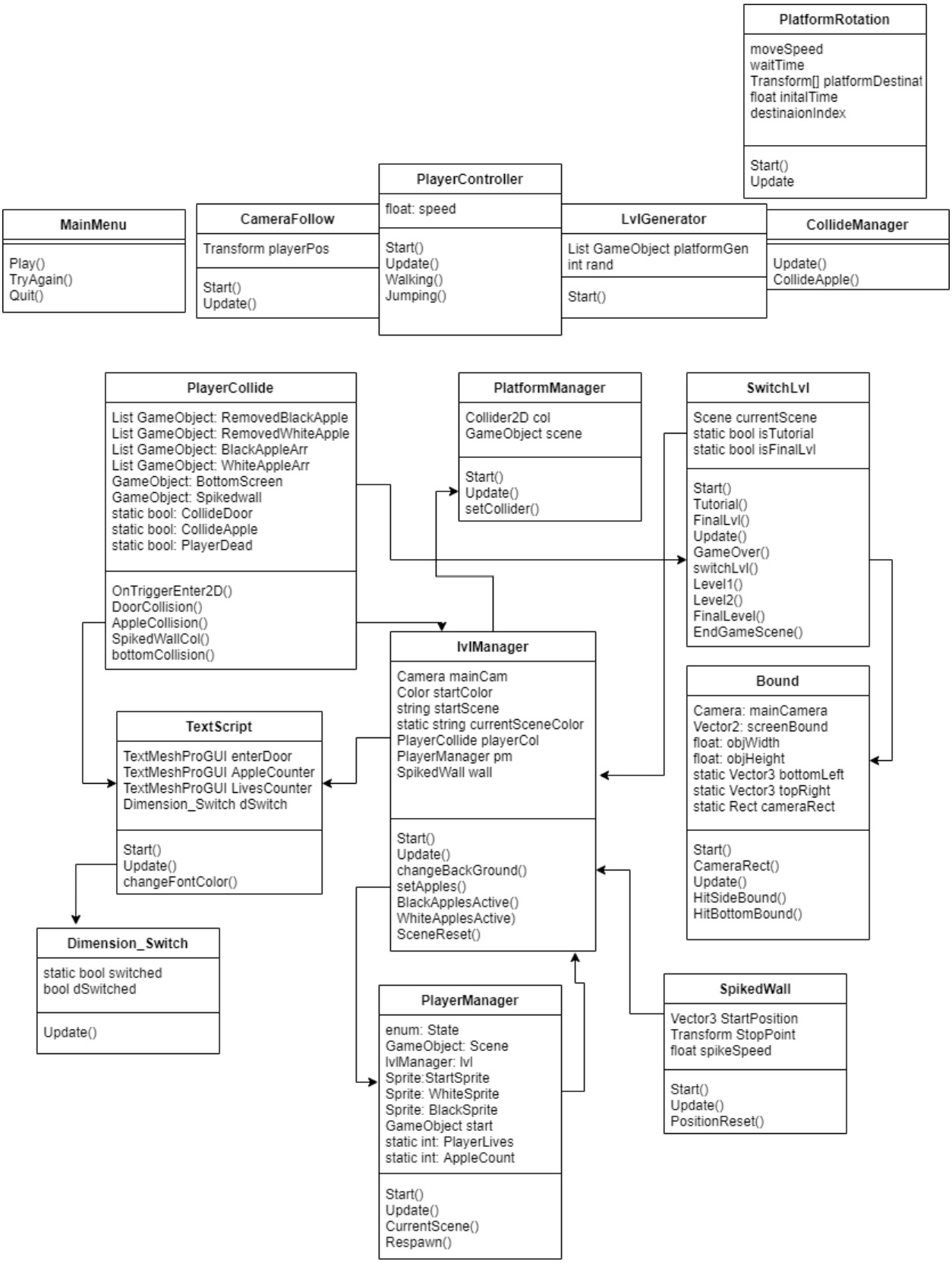
Postmortem – Game Programming

UML Diagram


PlatformRotation
moveSpeed
waitTime
Transform[] platformDestinat
float initalTime
destinaionIndex
Start()
Update
PlayerController
float: speed
Start()
Update()
Walking()
Jumping()
MainMenu
Play()
TryAgain()
Quit()
CameraFollow
Transform playerPos
Start()
Update()
LvlGenerator
List GameObject platformGen
int rand
Start()
CollideManager
Update()
CollideApple()
PlayerCollide
List GameObject: RemovedBlackApple
List GameObject: RemovedWhiteApple
List GameObject: BlackAppleArr
List GameObject: WhiteAppleArr
GameObject: BottomScreen
GameObject: Spikedwall
static bool: CollideDoor
static bool: CollideApple
static bool: PlayerDead
OnTriggerEnter2D()
DoorCollision()
AppleCollision()
SpikedWallCol()
bottomCollision()
PlatformManager
Collider2D col
GameObject scene
Start()
Update()
setCollider()
SwitchLvl
Scene currentScene
static bool isTutorial
static bool isFinalLvl
Start()
Tutorial()
FinalLvl()
Update()
GameOver()
switchLvl()
Level1()
Level2()
FinalLevel()
EndGameScene()
lvlManager
Camera mainCam
Color startColor
string startScene
static string currentSceneColor
PlayerCollide playerCol
PlayerManager pm
SpikedWall wall
Start()
Update()
changeBackGround()
setApples()
BlackApplesActive()
WhiteApplesActive)
SceneReset()
Bound
Camera: mainCamera
Vector2: screenBound
float: objWidth
float: objHeight
static Vector3 bottomLeft
static Vector3 topRight
static Rect cameraRect
Start()
CameraRect()
Update()
HitSideBound()
HitBottomBound()
TextScript
TextMeshProGUI enterDoor
TextMeshProGUI AppleCounter
TextMeshProGUI LivesCounter
Dimension_Switch dSwitch
Start()
Update()
changeFontColor()
Dimension_Switch
static bool switched
bool dSwitched
Update()
PlayerManager
enum: State
GameObject: Scene
lvlManager: lvl
Sprite:StartSprite
Sprite: WhiteSprite
Sprite: BlackSprite
GameObject start
static int: PlayerLives
static int: AppleCount
Start()
Update()
CurrentScene()
Respawn()
SpikedWall
Vector3 StartPosition
Transform StopPoint
float spikeSpeed
Start()
Update()
PositionReset()

I only had some minor dependencies for my game. A lot of it was just trying to access variables from other scripts so that something will trigger the code block of another script. A lot of the dependencies I used public *ScriptName* or using public static which seemed to be the most useful. However, a lot of things were not going right when I used a lot of the same static variables. Like some codes would not execute if an if-statement was true, but that was because it would turn false in another statement. So, getting certain things to work was very troublesome when a lot of static variables were being used. A lot of the time I had to get the variables by called the script when the gameObject.Find method, which worked, but only realized that you were not able to change the value, and also added a bit more work.

The class/components have no child base classes and is very standalone. For the game I made, I don't know if I could have added subclasses to any of the existing classes, but as the writer of the code. I am blinded from seeing much fault and a second observer could easily break down code and show how it can be done better.

I would honestly rate the re-usablility of my code a 2/10. A lot of the code is very specific for this game. I think the only code script that is not very specific towards this game is the player controller. Considering it's only consisting of movement that can be used mostly in other games. Probably another script would the lvlGenerator since it's taking prefab levels and storing them into an array for it to be later randomly generated to be used a level. This is probably another that can used in most games that want to take already existing level and randomly present it to the player. Other than that, everything else I think is specific to the game. The platform manager is designed so that it enables and disables certain platforms when the color switch is made. The PlayerManager has a state for changing the sprite depending on the color of the scene. LvlManager is designed to change the entire look of the level depending on if the switch is made. Even the Textscript requires specific variables from the game in order for it to work. All of which would probably only work in Unity. In my opinion, I don't see it as a Game Programming style of work, and it shows that I don't really know what it means to actually be a Game Programmer.

I think with how I at least styled my code. I can add any amount of Platforms and Apples as I wanted and it shouldn't trigger any problems as long as setting up the dependencies work correctly. I didn't have to make many adjustments when making the two levels for Level1. It was mostly drag and drop with connecting scripts to each other.

I did end up finishing my game with, I believe, everything I pitched in my proposal. There is a randomly generated level, albeit, randomly picks a generated level. There is also a "boss" level where a wall is chasing the player. The wall is a bit faster than the player, so the player doesn't have much room for error. So, I think I was able to be reasonable with the amount of time I had to work with. I don't know if I would redo my proposal, but I think I would definitely work on the code itself so that it doesn't have to be just a "code for this game only."