Retro game with a twist:

Super Wario Bros.

UML, activity diagram and game design patterns of this game by: game developer Mick Gerritsen

My game is based of the original Super Mario Bros game. It will take place in a place that looks like Las Vegas. Everything will control the same as the Super Mario bros game, except you play as Wario and some of the graphics will look different (as example, a coin block will look like a slot machine). The twist will be: Wario is hungry for coins. He wants to take as much coins as possible and those coins he puts in a bag. The fuller the bag will get with coins, the heavier Wario will get (so he can’t jump as high as without the coins he took). This will result in a game that feels like Super Mario Bros, but will be very puzzly: A mix of platforming and puzzling.

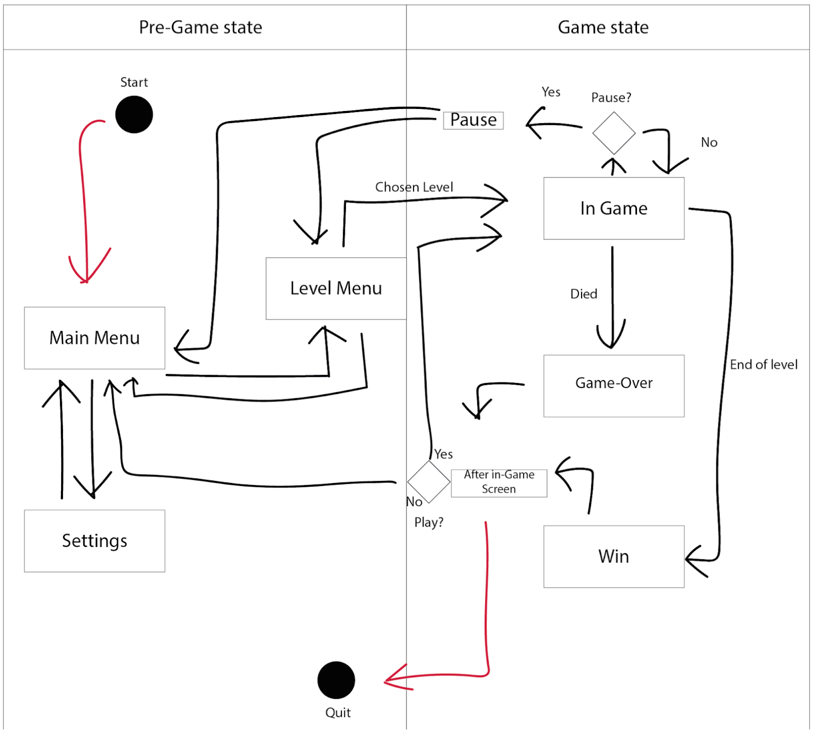
For the game I will use engine Unity 2018 because with this engine I feel the most comfortable of using it. While this engine might not be the best for 2D games, it will be a good option for me because I can easily make prefabs like blocks and enemies and there a few other things that will be handy when making a 2D game. This years version of Unity also comes packed with tools focused on 2D game development like the 2D tilemap tool, cinemachine and 9-slice sprite.

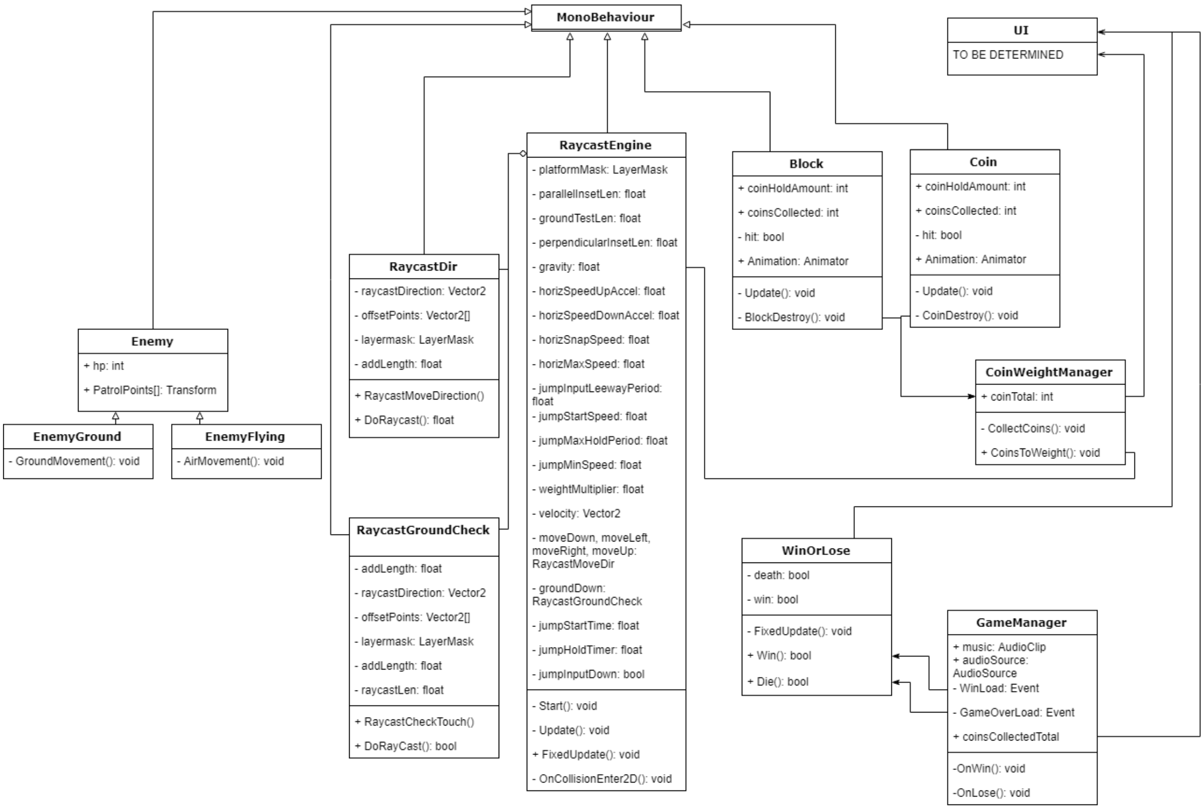
The code behind the game will be a composition. I will use raycasts and Unity’s build in 2d physics for the colliders and rigidbodies. To have great control over how the 2D movement of Wario will feel, I want to make a raycast playercontroller. With normal Unity physics it can also be done but I don’t want to do that because I want to make the players jump unique and control parameters that I otherwise can’t do unless I mess around with the 2D physics system in Unity that isn’t always as solid. For things as coins and items I will use colliders. The scripts on these gameobject will check for the collision with the player. I will also have a game manager script that will keep track of important things in my scene like the total amount of coins, music and game transition (Game Over or Win screen). I will also have coin weight manager that will convert the current amount of coins in one level to weight (gravity on player controller). This amount of current amount of coins will get passed over to a UI class as well as the all the coins collected in all levels and the weight of the player.

I will use a singleton for my game manager. The parts that need to be a singleton is the music because the music needs to player through the whole game. And I will use a singleton for the amount of coins the player collected through all levels. Because the player will hear the music and see the amount of collected coins on almost every screen of the game, I thought it was handy to code these as singletons.

For game-over or win I want to use events because I am very sure that there will be alot of things that need to work at the same moment. Because those will have seperate methods, I can put them in one event.

Object pooling is also something that I want to use. For blocks and coins I want to make an object pool. These are game objects that will appear or disappear in the game, because the in-game screen moves along with the player and the player will pick up coins. I also have a power-up in mind that looks like the fireball of Super Mario Bros. These fireballs can also have an object pool.


Pre-Game state
Game state
Start
Pause
Yes
Pause?
No
Chosen Level
In Game
Level Menu
Main Menu
Died
End of level
Game-Over
Yes
After in-Game Screen
No
Play?
Settings
Win
Quit


MonoBehaviour
UI
TO BE DETERMINED
RaycastEngine
- platformMask: LayerMask
- parallelInsetLen: float
- groundTestLen: float
- perpendicularInsetLen: float
- gravity: float
- horizSpeedUpAccel: float
- horizSpeedDownAccel: float
- horizSnapSpeed: float
- horizMaxSpeed: float
- jumpInputLeewayPeriod: float
- jumpStartSpeed: float
- jumpMaxHoldPeriod: float
- jumpMinSpeed: float
- weightMultiplier: float
- velocity: Vector2
- moveDown, moveLeft, moveRight, moveUp: RaycastMoveDir
- groundDown: RaycastGroundCheck
- jumpStartTime: float
- jumpHoldTimer: float
- jumpInputDown: bool
- Start(): void
- Update(): void
+ FixedUpdate(): void
- OnCollisionEnter2D(): void
Block
+ coinHoldAmount: int
+ coinsCollected: int
- hit: bool
+ Animation: Animator
- Update(): void
- BlockDestroy(): void
Coin
+ coinHoldAmount: int
+ coinsCollected: int
- hit: bool
+ Animation: Animator
- Update(): void
- CoinDestroy(): void
RaycastDir
- raycastDirection: Vector2
- offsetPoints: Vector2[]
- layermask: LayerMask
- addLength: float
+ RaycastMoveDirection()
+ DoRaycast(): float
Enemy
+ hp: int
+ PatrolPoints[]: Transform
EnemyGround
- GroundMovement(): void
EnemyFlying
- AirMovement(): void
CoinWeightManager
+ coinTotal: int
- CollectCoins(): void
+ CoinsToWeight(): void
RaycastGroundCheck
- addLength: float
- raycastDirection: Vector2
- offsetPoints: Vector2[]
- layermask: LayerMask
- addLength: float
- raycastLen: float
+ RaycastCheckTouch()
+ DoRayCast(): bool
WinOrLose
- death: bool
- win: bool
- FixedUpdate(): void
+ Win(): bool
+ Die(): bool
GameManager
+ music: AudioClip
+ audioSource: AudioSource
- WinLoad: Event
- GameOverLoad: Event
+ coinsCollectedTotal
-OnWin(): void
-OnLose(): void