Kat Stleva Nyah

Software Engineering Document

Table of Contents

[Game Mechanics 1](#_Toc510635067)

[Game Scenes 1](#_Toc510635068)

[Game Flow 1](#_Toc510635069)

[Classes and Scripts 1](#_Toc510635070)

# Game Mechanics

## Game Scenes

The game will have four basic scenes:

* A start scene, where we will allow the user to start from the first level or to quit the game.
* A series of five game scenes, where the player will explore and recollect items, as well as avoid falls and certain enemies.
* A win scene, where the player will get to when and if they complete the fifth level, they will be shown their final score and the level they reached.
* A lose scene, where the player will get to when they lose their 3 lives, or if they fall from any level.

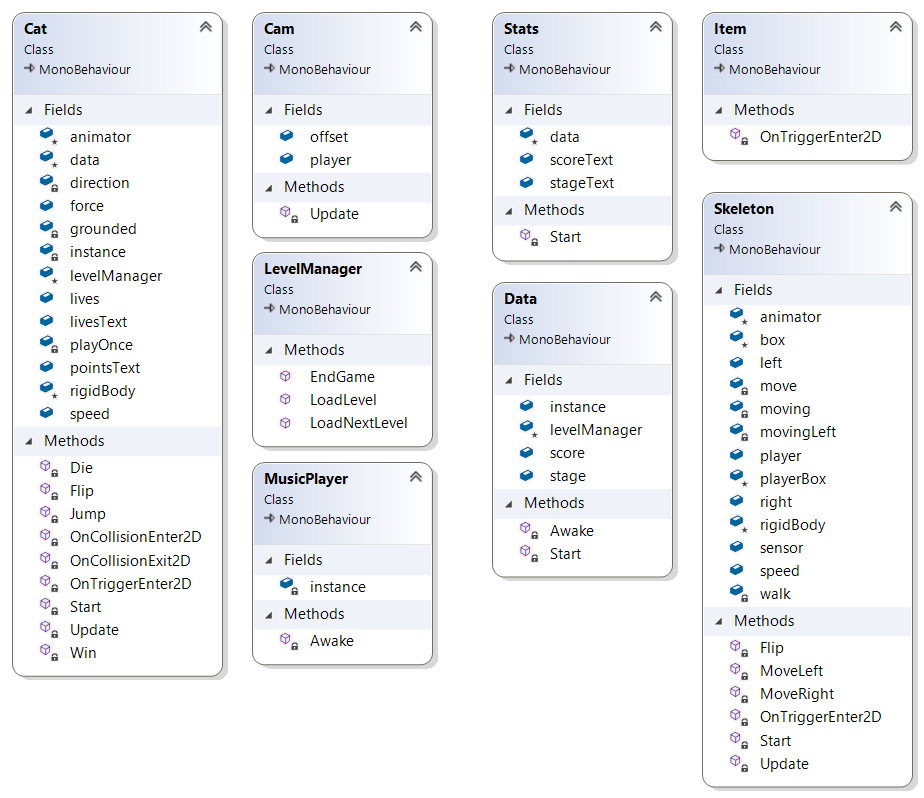
## Game Flow

* Start scene will lead to the level 1 of the game.
* Both the Win or Lose scenes’ will lead back to the Start scene.
* The game scenes will lead to:
  + The next level when and if the player, reaches a golden yarn ball.
  + Win scene, when and if the player reaches a special item in the fifth level.
  + Lose scene, when and if the player reaches zero lives or the player falls from any level.
* The Start scene will also let the player choose to quit the game.
* Both the Win or Lose scenes’ will also let the player choose to quit the game.

# Classes and Scripts

We handle a set of objects throughout the game:

* A LevelManager, that allows us to move from one scene to another.
* A MusicPlayer, that will allow us to put background music to our game.
* An Item, will be instantiated several times throughout the game and allow us to keep track of the collectibles in our game.
* A Cam, that allows the camera to follow the movements of the player on the x-axis.
* Data, allows us to keep track of the total score, and the level reached.
* Stats, allows us to update the texts’ that are displayed to us with the total score and the stage reached.
* A Cat, which is the main character that the player controls, it is instantiated every Game scene throughout the game.
* A Skeleton, which is used by both the skeleton and ghost enemies, this means we’ll have the same behaviors for both enemy types. Instantiated many times throughout the game in each Game scene.



LevelManager:

* LoadLevel(name), calls for a scene with the SceneManager by its name.
* EndGame(), will terminate the game.
* LoadNextLevel(), calls the next scene in the build settings order.

MusicPlayer:

* Attributes:
  + Instance, the creation of the same music player.
* Methods:
  + Awake(), uses the singleton algorithm, lets us initialize the MusicPlayer.

Item:

* OnTriggerEnter2D(Collider2D collision), when the item touches the player, the item will destroy itself.

Cam:

* Attributes:
  + Player, the transform position of the player.
  + Offset, the position of the camera.
* Methods:
  + Update(), transforms the position of the camera so that it follows the player position in the x-axis.

Data:

* Attributes:
  + Instance, the creation of the same Data object.
  + levelManager, the instance of the LevelManager object.
  + Score, the counter of the total accumulated score.
  + Level, the counter of the level reached.
* Methods:
  + Start(), initializes the levelManager instance and sets the counters of the score and level to 0 and 1 respectively.
  + Awake(), uses the singleton algorithm , lets us have one instance of the Data object.

Stats:

* Attributes:
  + Data, an instance of the Data object.
  + scoreText, text object used to update the total score.
  + stageText, text object used to update the level reached.
* Methods:
  + Start(), initializes the Data object to have the data stored, and sets the text attributes to the corresponding values so they can be viewed in the level, after the data has been shown, set the score and level of the Data object to 0.

Cat:

* Attributes:
  + Instance, the same instance of the cat object.
  + Speed, used for the walking force transform.
  + Force, used for the knockback force that the player experiences.
  + Lives, keeps track of the current player lives.
  + livesText, the text used in the canvas to update the total lives.
  + pointsText, the text used in the canvas to update the total score.
  + levelManager, the instance of the LevelManager of the game.
  + rigidBody, the instance of the RigidBody2D used throughout the game.
  + animator, the instance of the animator component.
  + data, the instance of the Data object.
  + grounded, used to check if the player is touching or not the floor.
  + direction, used to modify the direction the player is facing.
* Methods:
  + Start(), used to initialize the various components used in the game; rigidBody, animator, levelManager, the texts objects, the data instance.
  + Update(), used to check the various keys the user inputs and reflect its with the animator of the Cat character; moving, jumping or ducking.
  + Flip(), method to flip the sprite to the opposite side.
  + OnTriggerEnter2D(Collider2D collision), checks if the player collects any items, and add points to it’s final score, also checks if the player touches the fall collider to take him to the Lose scene.
  + Jump(), coroutine that sets the necessary value for the animator to do the jumping animation.
  + OnCollisionEnter2D(Collision2D collision), checks if the player touches an enemy and sets the values to execute the hurt animation, diminishes the cats lives and adds a knockback force to the player. Also if the player is touching the ground set the grounded boolean to true.
  + OnCollisionExit2D(Collision2D collision), checks if the player leaves the floor and sets the grounded Boolean to false.
  + Die(), destroys the player, and loads the Lose scene.
  + Win(), destroys the player, and loads the next scene in the build order.

Skeleton (used by both the skeleton and ghost enemies):

* Attributes:
  + Speed, used for the walking force transform.
  + Sensor, an instance of a BoxCollider2D that detects the player.
  + Player, an instance of the BoxCollider2D of the player.
  + Left and Right, instances of the walls that restrain the walking behavior of the enemy.
  + movingLeft, Boolean that lets us know in which direction the enemy is facing.
  + walk, vector3 object that lets us transform the enemy position.
  + moving, boolean that lets us know if the the enemy is moving or attacking.
  + rigidBody, the instance of the RigidBody2D component.
  + animator, the instance of the animator component.
  + box, the instance of the sensor BoxCollider2D.
  + playerBox, the instance of the player’s BoxCollider2D.
* Methods:
  + Start(), initializes the rigidBody, animator, box, and playerBox variables.
  + Update(), checks if the sensor is touching the player and if so begin the attacking animation, else checks if the moving variable is true to begin moving either left or right with the corresponding methods.
  + Flip(), used to flip the enemy’s sprite to the opposite side.
  + MoveLeft(), transforms the position of the enemy so it moves left, calls the flip function, sets the movingLeft boolean to true, and the changes the walk.x direction to -1.
  + MoveRight(), transforms the position of the enemy so it moves right, calls the flip function, sets the movingLeft boolean to false, and the changes the walk.x direction to 1.
  + OnTriggerEnter2D(Collider2D collision), if the enemy touches the Wall object sets the movingLeft boolean to false or else to true.