# **Lumber-Man**

*The Space Paradox*

Iftime Adrian-Dumitru - 1207B

Gameplay: Single player missions where the player must collect coins to open a portal for the next level. Gameplay involves traversing the levels, avoiding the toxic water and slaying the monsters that stand in the way to the portal. The player can avoid the monsters and still get to the next level but by killing them, he receives bonus score points. On top of that, the more health points he as at the of a level, the more points he gets. If the player touches the toxic water or if his health points drop to 0, the player dies. Every time the player gets hit, he becomes invulnerable for a brief period.

*Plot:* The action happens in the Promethia Galaxy on a number of different planets. A simple Lumber-man is swept away from his home through a portal and left alone in a dangerous swamp. The protagonist searches a series of magical coins that open rifts hopping one will lead him to Earth.

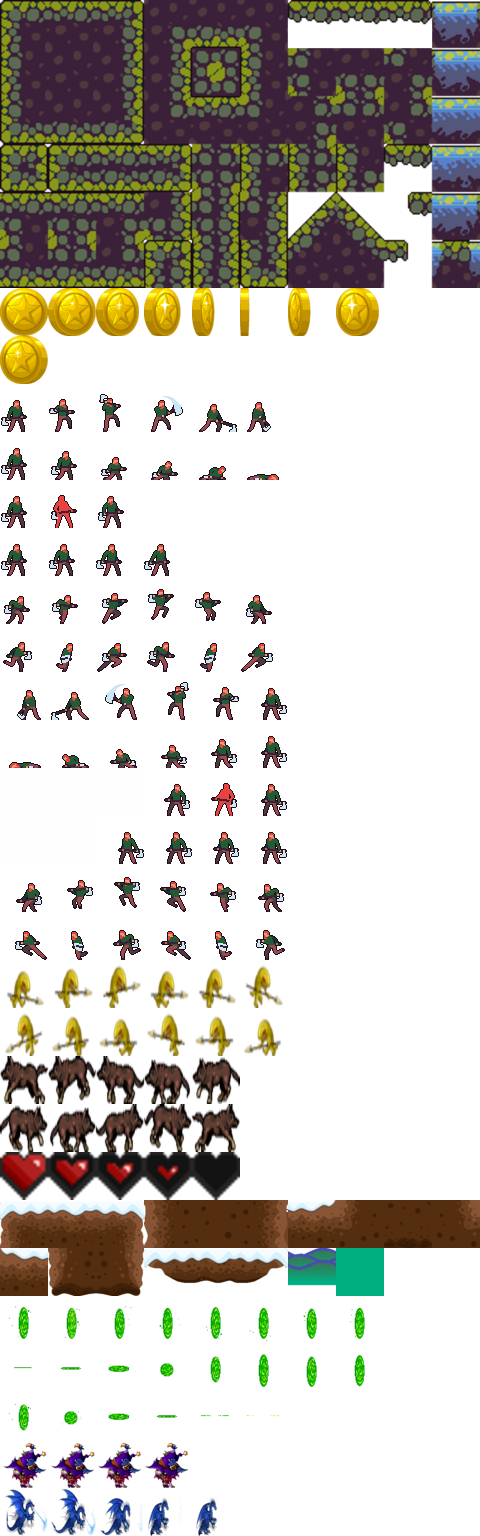
*Characters:*

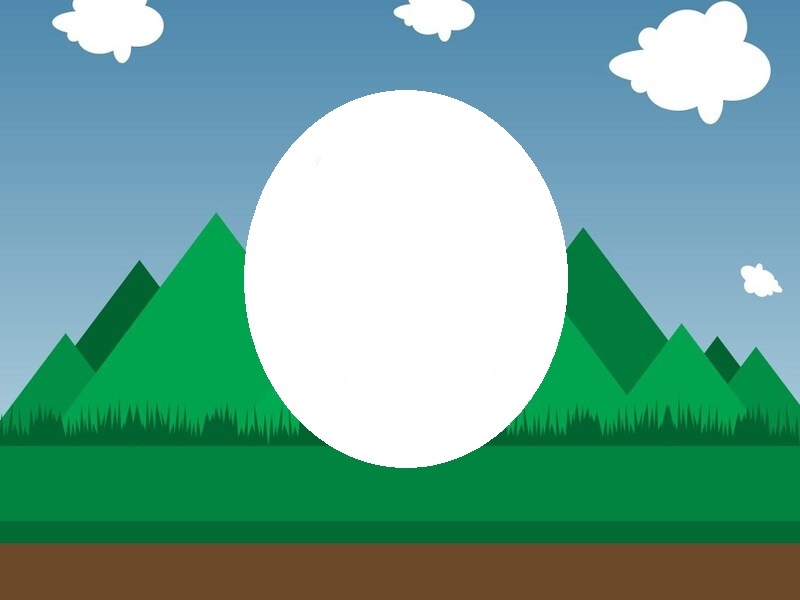
* The Lumber-Man is the main character of the game. He doesn’t possess any supernatural power and is portrayed as a normal human being. His only way of protecting himself is by using his raw power and trust-worthy axe.
* The fighting dummy was created by the protagonist. It was built with the intention of training.
* The monsters consist of big timber wolves and agile lizard men. They are hostile to the protagonist and will attack if he is in their melee range.
* The Ice Dragon is the final boss of the game. He is almost 3 times the size of the lumber-man and has 2 type of attacks: normal attacks in melee range and Icy Breath if the player is outside of melee.

*Mechanics:*

* Game-points: The player receives points by doing a list of actions: avoiding the monsters, collecting coins slaying beasts and also by advancing to the next level.
* Controls: The movement controls are w a s d or the arrows, enter to select a button in the menu and space to attack. The movement controls can be swapped in the settings menu.
* Wall-Running: The player can run on walls to get to more difficult places by jumping to a wall then changing the facing direction in the opposite direction and pressing the jump button again.
* Saving the game: The player can press “p” at any time while in a level and the game will register the level and the score he had when starting the current level. After that the player can go to the load section of the main menu and continue from the saved level.
* Health-points: The player starts the level with 4 health points. These points reset at the start of every level. If the points reach 0, the player is dead and the player is sent to the starting menu.
* Collision and falling: The player is stopped if he tries to move through solid blocks and also the player starts falling down if he doesn’t have any blocks beneath him.

*The Sprites:*

* The sprites were all collected from the Internet and edited to have the correct size and then joined in a big sprite-sheet. The backgrounds and the dragon sprites kept their size because they had to have a better resolution.



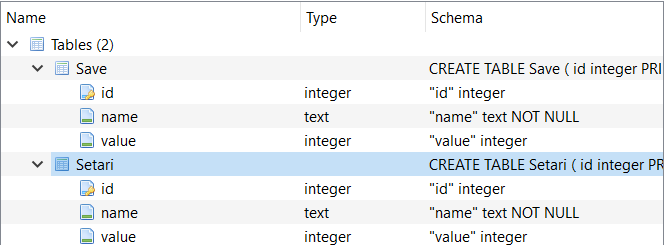


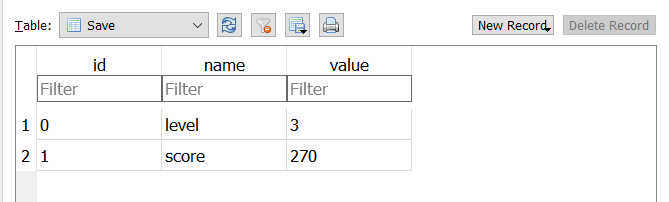


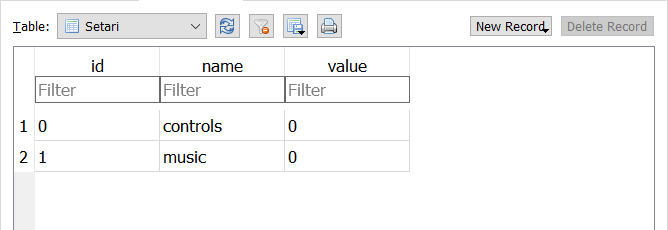


*The database:*

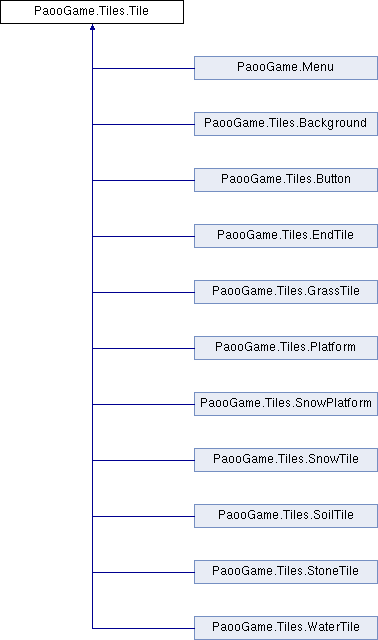
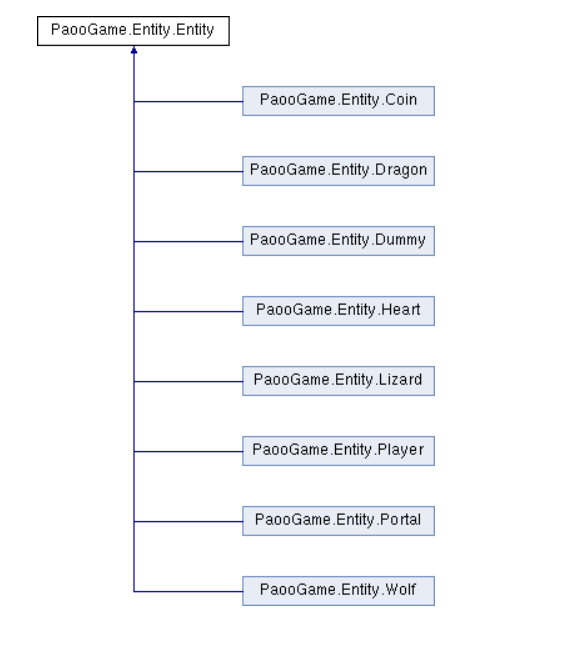
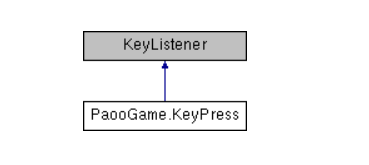
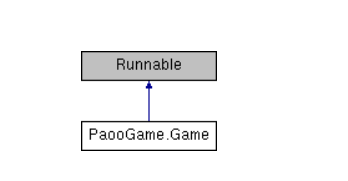
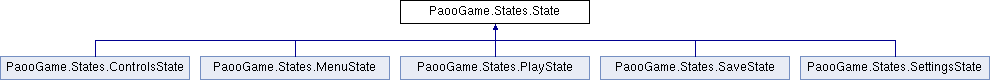
* Consists of 2 tables:
  + - one table keeps the current settings: Music ON/OFF and the control buttons
    - one table keeps the level and score of the last saved game



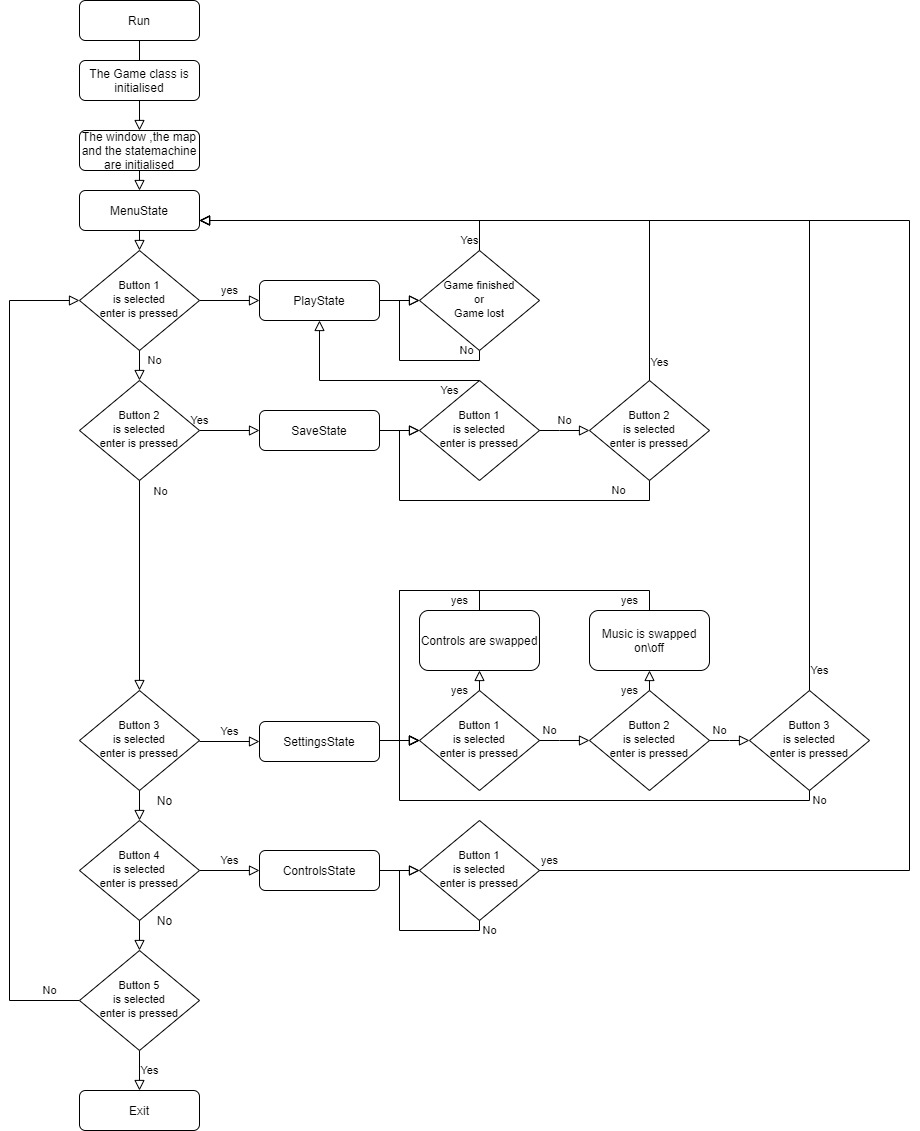




*Class diagrams:*



*The program’s flowchart:*



*Descriptive technical documentation:*

The game works following the next pattern: when the main function is called a Game instance will be created. In the Game constructor the program will create the map and set it to level 1, it will create the window in which the game will be drawn and also it will set the state machine to Menu-State to display the menu. When any of the menu buttons will be selected, the state machine will be set to a different state. If the start button is selected the design of the level will start to be take shape: the background will be drawn, on top of the background there will be the tiles and also the entities. After collecting all the coins in a level and reaching the portal, the level will change, the map will be swapped with the next one. If the player finishes the game or dies, the state-machine will be set to menu.

The Game class is one the most important classes because it creates the other objects and also calls the draw and update functions of those objects. If the user selects the settings menu or he loads the game the program will interrogate the database and will read from the tables the mandatory data. If the player saves or changes the settings the program will update the tables and save the database.

To make the game run smoother the Game class extends runnable so the tasks are run at the same time. This fact helps the map creation a lot because without it, the game would have to wait for every tile and entity to get drawn and updated. If while in the menu, the exit button is pressed then the stopGame function is called: the runState parameter is swapped to false and the window closes. To know when a keyboard button is pressed the program uses the KeyPress class. The music of the game is being played using the MusicPlayer class which reads the song file decodes it and then plays the audio.