**Labyrinth Escape**

A video game by Jorge Leonardo Quimi Villon

Computer graphics arch exam – games and simulation

**Target**

Create a game, in this case rework the labyrinth covered during the lessons.

**Introduction**

The development of the project was carried out using the Monogame framework and Visual Studio taking advantage of the knowledge learned during the COMPUTER GRAPHICS ARCH.- GAMES AND SIMULATION course and deepening the concepts necessary for the realization.

The name of the game is “labyrinth escape” and the main purpose is to find the exit path.

The player can move the character into the labyrinth where the key must be found to open the door at the end of the level. Along the way there will be obstacles that must be avoided, Otherwise the character will have to start the current level from the beginning and lose life. When the character has finished all the available lives there will be GAME OVER.

The game contains a timer which keeps the time taken to finish the levels in a single match. Finally, a comparison is made between the current time and the best time. In case, the current time is less than the best time, the current time takes the place of the best time.

The game is divided into several levels, in increasing order of difficulty.

**MonoGame framework**

Labyrinth Escape was create using MonoGame, an open Source framework for cross-platform videogames: it allows videogames porting to many platforms including Windows, Linux, iOS and Android. MonoGame allows the development of videogames by means of the C# object programming language by offering numerous classes that contain indispensable tools.

The structure of a game created with MonoGame is essentially based on three classes:

* *Game:* the heart of the videogame which is also the entry point of the program. It contains the interfaces of the essential methods for the game and called by the base class:
  + *Inizialize():*  contains the basic initialization of the graphics and input sector;
  + LoadConten(): upload multimedia content;
  + *Update():* method called before rendering of each frame. It takes as input a GameTime object used to get a lot of information about the game's time status
  + Draw(): renders all the elements of the game on the screen. As Update(), it takes a GameTime object as argument
* Graphics.SpriteBatch: allows the rendering of all Texture2D objects
* Content.ContentManager: allows to manage all the multimedia resources of the game. For this purpose, the framework provides the Content Pipeline tool which, by means of a graphic interface, allows an easier management of the contents.

**The Structure of Recycle Game**