CHANGES FROM THE ORIGINAL PROPOSAL

ORIGINAL PROPOSAL	END RESULT
Maze generator with random difficulty of the levels.	We create a maze generator that every time you run the program, it changes walls, there is a certain path that is always the same, but there are other path create randomly.
The player is a bouncing ball that goes from starting point A to ending point B through the maze.	The player is a bouncing ball that goes from starting point A which is the first cell with coordinates x[0,100] and y[0,100] to ending point B with coordinates x[900,1000] y[900,1000] through the maze. Another fetch added is that the ball can go through the maze from a position to another by exceed the sided walls.
It moves thanks to the function big-bang, it moves affected by the gravity and bounce when there's a wall.	It moves with the event on-tick of the function big-bang. The ball moves as if it's in the physic world: it's affected by gravity and decelerates itself if the user don't press any arrow; it accelerates and decelerates if the user press the "left" and "right" arrows; if the pressing arrow is the "up", the ball goes up while "down" it stops or goes down. If the ball touches the walls the velocity is inverted and the ball bounces in the other opposite direction.
We define a function for big-bang that will have full screen, an handle key, handle mouse on tick	The function big-bang have the display mode of full screen, to draw, on tick and on key.
Pressing "esc" quit the application's window.	Pressing "esc" quit the application's window.
Pressing the arrow, the player will move.	Pressing the "up" the ball accelerates to up, pressing "left" it moves to left or it decelerates from right, pressing "down" it decelerates from up, pressing "right" it moves to right or it decelerates from left.
Pressing a button that we will decide the player return in his initial position.	Pressing the "r" the ball returns to the original position that is the first cell.
We will use the racket language	We use the advanced language and not the entire racket language.

Construct the maze with structs, scene+line, make-pen, empty-scene and more.	We construct the maze with structs, constants, functions, recursions, world, scene+line, place-image, add-line make-pen, empty-scene.
Create a timer that starts when the game begins.	We decided to not put the timer because the language that we studied doesn't allow us to implement this, otherwise we would have used the entire racket language and not the advanced.
If we have enough time we will develop more levels.	We don't develop more levels because we make sure a path for every time that the user run the maze.
Option SURVIVAL MAZE	We don't implement the survival maze because we have done the first option that is the real game since the origin.