REQUIREMENT SPECIFICATION

1. FUNTIONAL REQUIREMENTS
   1. Play Game

Car Game is based on a simple car game which is enriched by some arcade features in order to entertain the user. Initially, user has 3 healths, appears on a 3 line road. Randomly generated other enemy cars eventually appears randomly on the road. Main purpose of the user is escaping from enemies by using left and right arrow and getting the highest score. User gets points for each second passed alive. Also player will encounter some extras as obstacles and power ups, such as stronger car, water holes, black holes.

* 1. Change Settings

User have chance to change the user’s car. There are multiple choices for the user’s cars in terms of design of the car. There are no extra’s about selecting car by us.

* 1. High Score

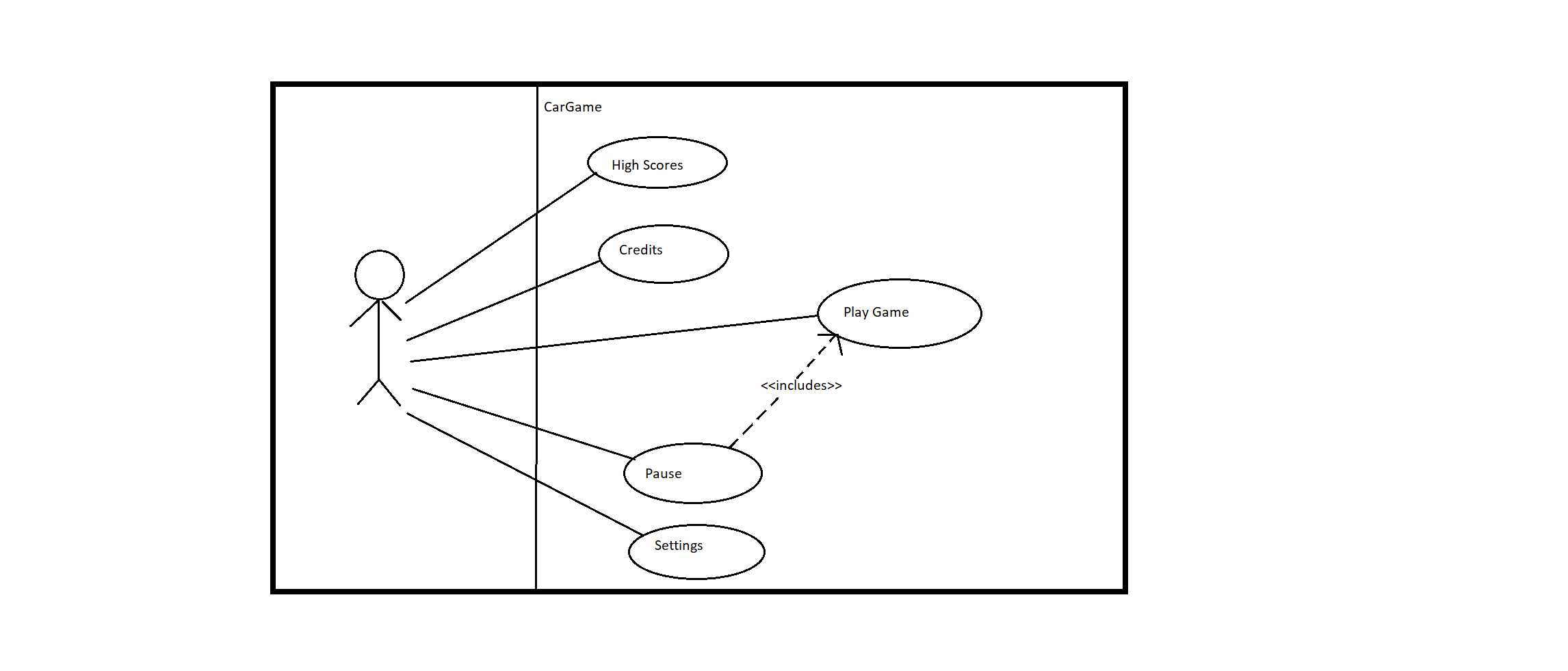
Player can see previously recorded highest records, and user cannot see his previous records unless high score is among in the highest scores.

* 1. Pause Game

The game can be stopped by the user, in the pause state, game stops until user starts it again.

* 1. Credits

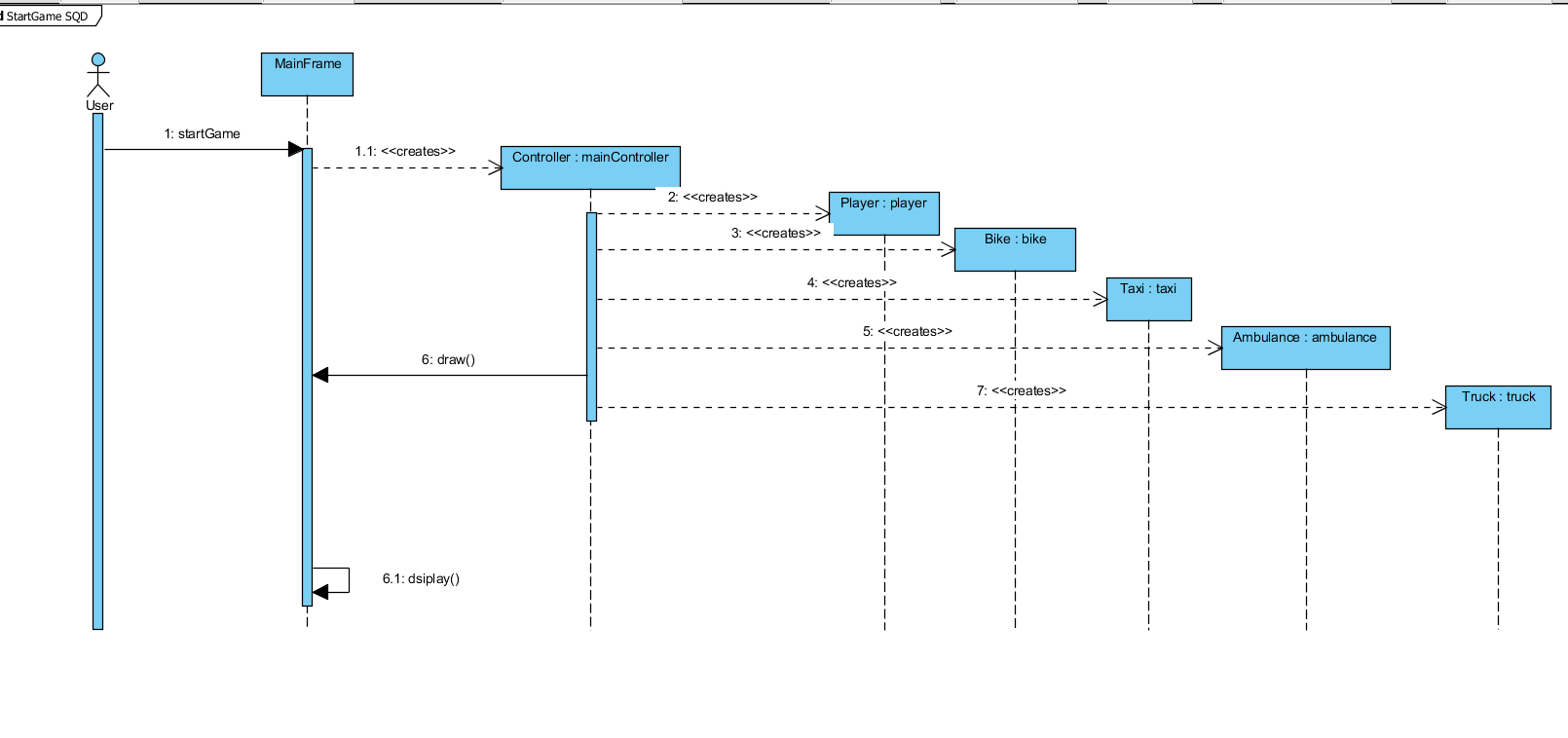
Gives information about the creators of the program.

User

1. Use Case Name : Show High Scores
   1. Actor : User
   2. Flow of events :
      1. User clicks “High score” button.
      2. System shows the top scores.
      3. User returns to main menu.
   3. Entry Conditions
      1. User clicks “High score” button at Main Menu
   4. Exit Conditions :
      1. User clicks “Main Menu” button
   5. Quality Requirements : NONE
2. Use Case Name : Pause Game
   1. Actor : User
   2. Flow of events :
      1. User clicks “Pause” button during the gameplay
      2. System stops running.
   3. Entry Conditions
      1. User clicks “Pause” button during the game.
   4. Exit Conditions
      1. User clicks “Continue” button OR
      2. User closes whole program
   5. Quality Requirements : NONE
3. Use Case Name : Change Settings
   1. Actor : User
   2. Flow of events
      1. User clicks “Settings” button.
      2. System shows the modifiable preferences.
      3. User can change the car in terms of color, shape.
      4. System saves modifications
      5. User returns main menu
   3. Entry Condition
      1. User clicks “Settings” button during
   4. Exit Condition
      1. User clicks “Main Menu” button
   5. Quality Conditions : NONE
4. Use Case Name : Play Game
   1. Actor : User
   2. Flow of Events :
      1. User send message to operating system in order to play game
      2. System opens the game
      3. Main Menu opens.
      4. User starts playing the game
      5. System keeps running the game until user lose his all healths
      6. If user’s score is in top five score, program records the score as High Score.
      7. User returns to Main menu.
   3. Alternative Flow of Events :
      1. User decides directly close the game ( go step 5.2.7 )
   4. Entry Conditions
      1. User runs the game.
   5. Exit Conditions
      1. User closes the program.
      2. User loses all health points.

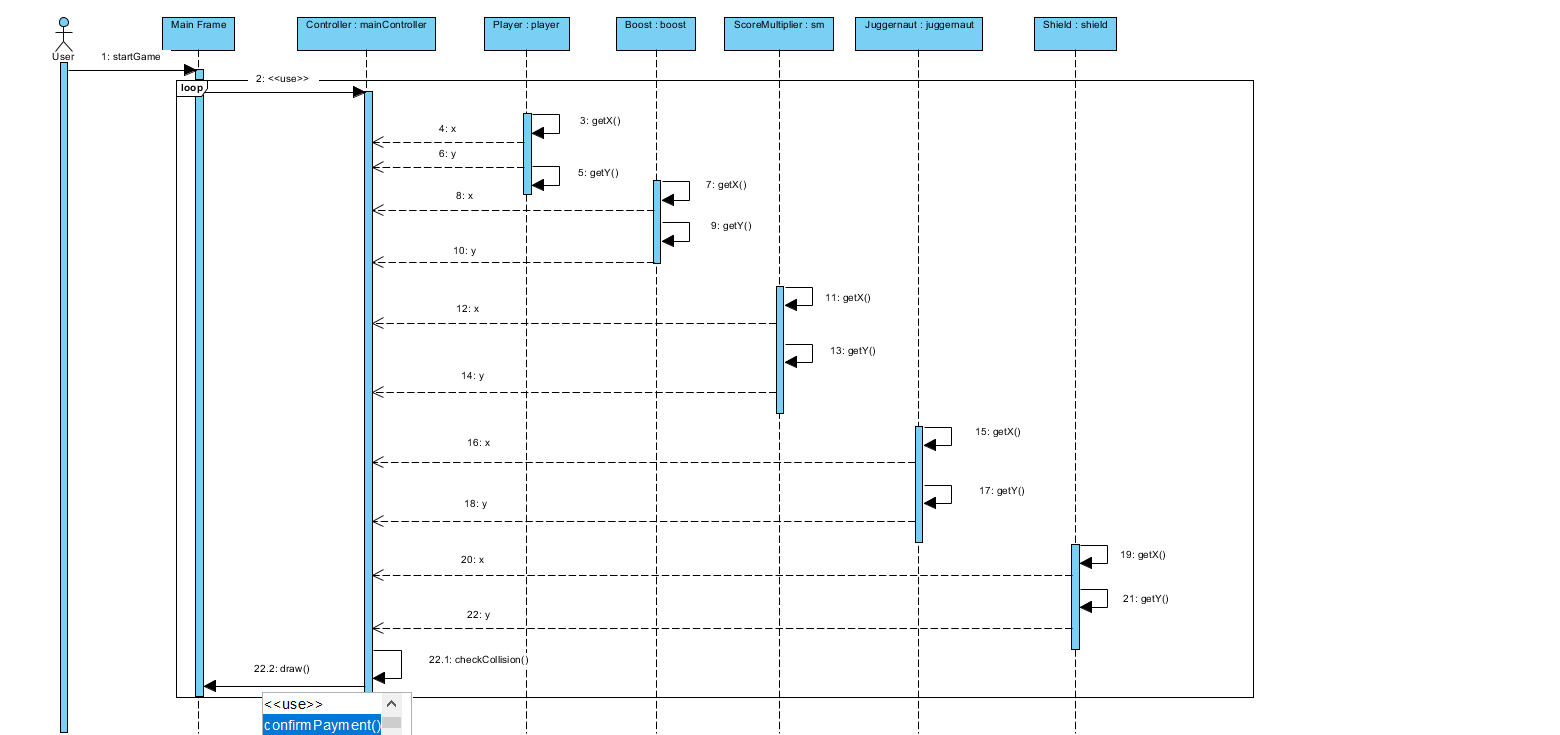
**Start Game**

**Scenario :** Player invokes proper method to initialize Main frame of the game, which main menu of the game. When main frame initialized, player clicks start game button in order to initialize game. After initialization of the game, Main Frame creates a Controller object, which controls the object of the software mainly. After controller object created, a bike, taxi , ambulance and truck objects are created as enemies by the controller, and player object creates by controller as well. After all enemy and player objects created, those objects are drawn on to main frame by the controller. Lastly, Main Frame invokes display methode in order to represent graphics on the screen.



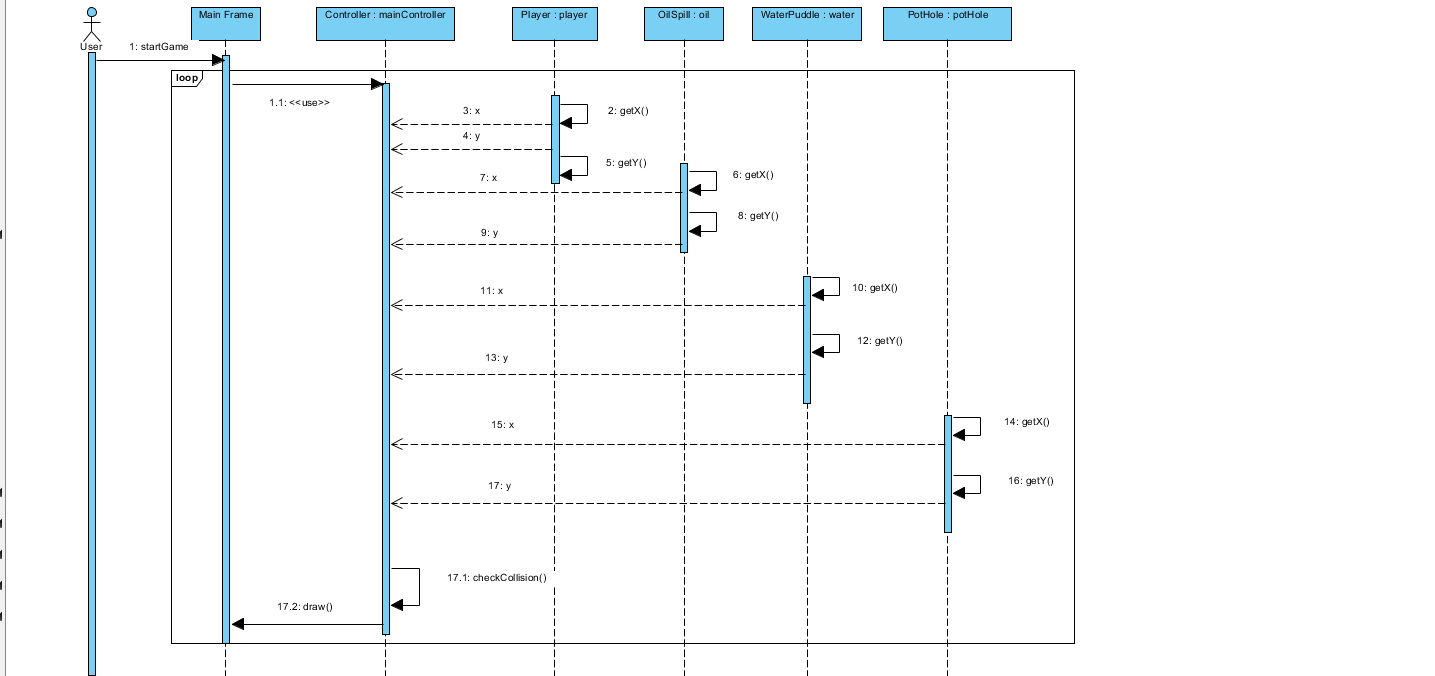
**Power Up Mechanism**

**Scenario :** User request to initialize Main frame of the game. Assuming previous diagram performed. After game initialized on the screen, in other words, user starts playing the, system enters game play loop which performs game dynamics. Assumed that user moves the player icon on the screen in order to escape from enemies. In the game, we assumed that all power ups created after game initialized and all four powers can be on the screen at the same time. If system generates a power up on the main frame, controller must know the coordinates of emerged power up. When user’s x and y coordinates matches with the power up’s x and y coordinate, user powers up according to which power up that user collided with. If a user powers up, controller draws the power up effects on the main frame



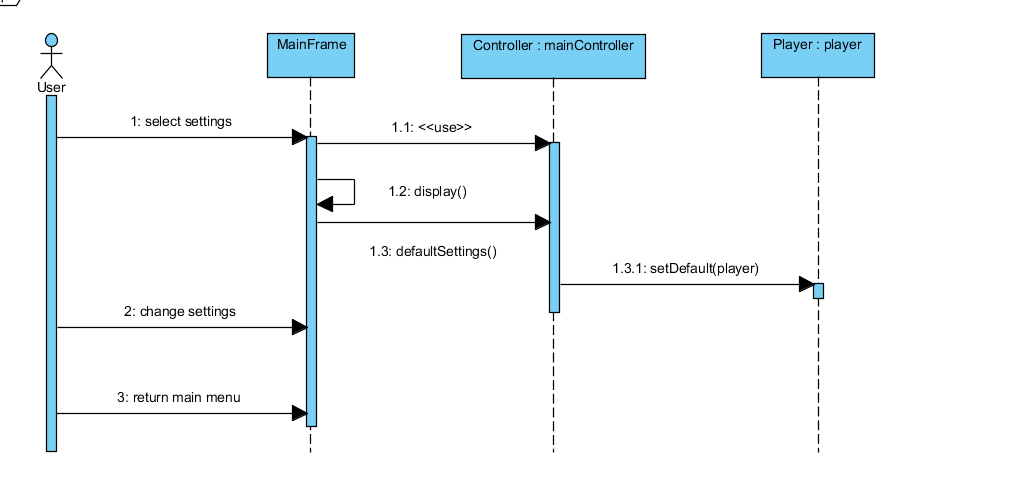
**Obstacle Mechnanism**

**Scenario :** User request from operating system to invoke the game. Assuming Start Game case diagram performed. After game initialized on the screen, obstacles generated randomly as time goes by according to game. There are four obstacles for the game, and they can be on the screen at the same time. Obstacles are generated randomly by the system on the screen. When an obstacle or obstacles emerge on the screen, controller must know generated obstacles coordinates in terms of x and y. If player’s coordinates match with the obstacle’s coordinates, user will be affected according to obstacle that user collides with. Controller draws the effects of the obstacle on the player to the main screen.

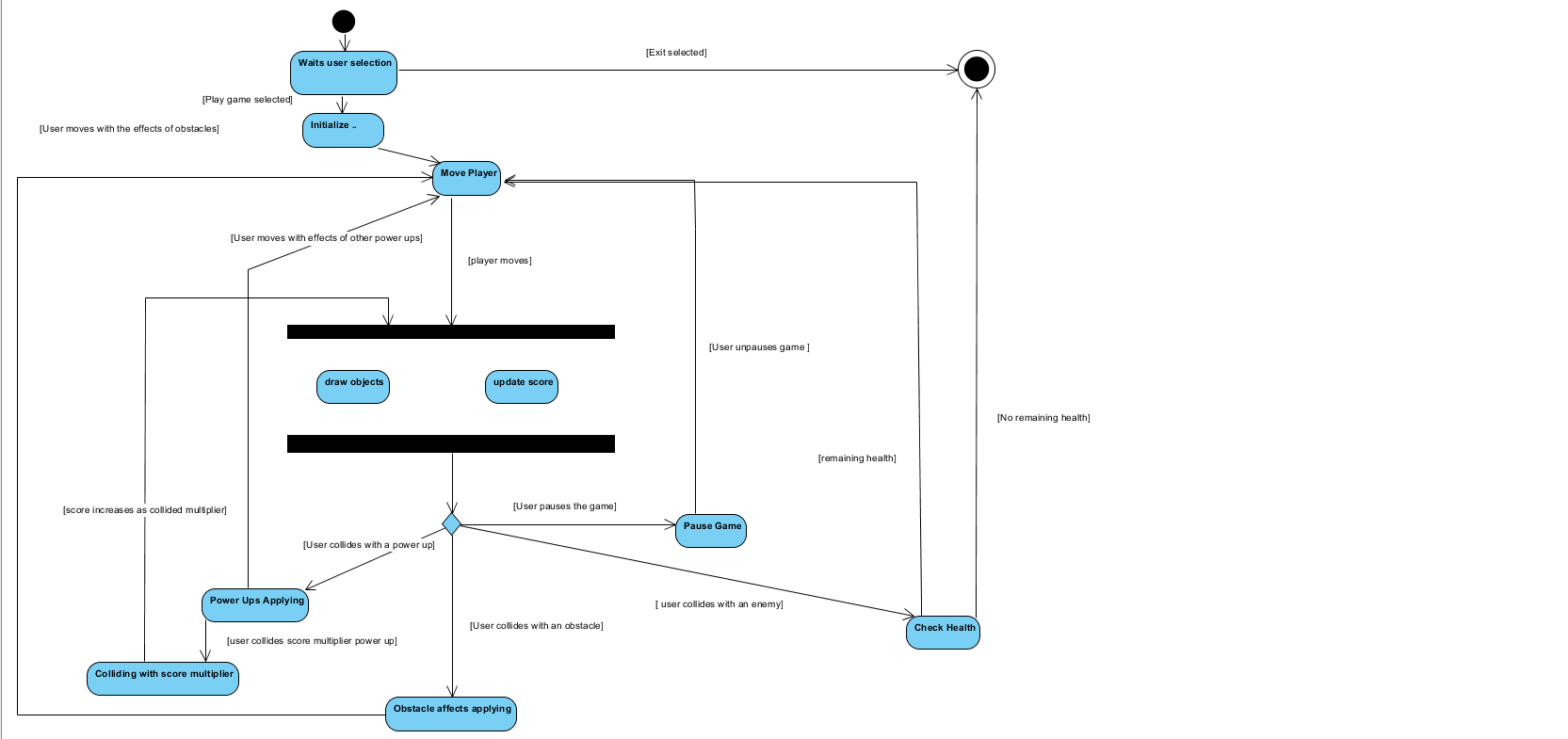


**Change Settings**

**Scenario :** User requests game invocation from operation system. Assuming Start case diagram performed. After initialization of the Main frame on the screen, user selects settings on the main menu which is shown by Main Frame. User selects settings on the main menu.Then, system initialize the players current preferens as defaults. User can change the icon of the player from this screen. After returning main menu, system records alterations on the player.



**Activity Diagram**

****

When game is run by operating system, program awaits user to receive a signal. If received signal is exit game, then program closes. If user’s signal is play game, then program start initializes every aspect of the game; player, controller, obstacles, power ups, enemies and so on. After initialization, enemies start coming against the player, and the player tries to avoid colliding with enemies. If player collides with an enemy, player’s health point reduces. As player increase its score during the game, frequency of emerging enemies and obstacles will increase, but power ups will remain with the same frequency as game starts.

Also, power ups and obstacles are generated by the system on the screen with random places. If user collides score multiplier power up, increasing rate of the score will be multiplied by 2. Collision with other power ups will not affect the score increase directly, but gives advantage to user, such as shield, which ignores one collision with an enemy, juggernaut, which destroys an enemy after collision, and boost, which makes player move faster in horizontal axis.

Obstacles are like power ups in terms initializing, emerging on the screen, but they are disadvantages for player. If player collides with water puddle, movement of horizontal axis would deviate from normal horizontal movement as moving not with certain units. Oil Spill has similar aspects, but the movement deviation is much bigger than water puddle. Pot Hole is different kind of obstacle, if player collides with a pot hole, one of his health point will reduce.

The game will go on as long as player keeps its health points, if player have no health points left, then system reports that game is over and then returns user to the main menu. Also user can quit program by directly closing it.