# Design Overview for Chicken Invader

Name: Dang Vi Luan

Student ID: 103802759

# Summary of Program

Growing up during the 2000s, I have been playing video games ever since, and throughout my childhood, I have been fascinated by the famous shoot ‘em up video game called Chicken Invader. For the higher distinction task of week 9, I plan to recreate the first version of this nostalgia game series.

The main theme of the game is the battle to protect the Earth against an advanced race of space-faring chickens, who are planning on subjugating and enslaving human beings. The player will control the battleship to defend against the attack of the chickens.

The player will lose if any of the chickens cross the program border, the player will also lose if the ship collides with the chicken. After shooting down 100 chickens, the invasion will be stopped and the Earth will prevail.

Initial sketch of the program:

A screenshot of a computer

Description automatically generated with medium confidence

# Required Data Types

Describe each of the records and enumerations you will create using the following table (one per record).

File1: Main.rb

|  |  |  |
| --- | --- | --- |
| Field Name | Type | Notes |
| Width, Height | Integer | Width and Height of the program |
| Xy\_values | Array | Array used to store enemy in preformatted order |
| @enemy | Array | Array used to store chickens |
| @player | Object | Generate player |
| @bullet | Array | Array used to store bullet |
| @explosion | Array | Array used to store explosion |
| @credit | Array | Array used to store credit |
| @font | Gosu::Font | Standard font size of the whole program |
| @background\_image | Gosu::Image | Background image |
| @start\_music | Gosu::Song | Start music |
| @shooting\_sound | Gosu::Sample | Shooting sound |
| @explosion\_sound | Gosu:Sample | Explosion sound |
| @message | String | Credit message |
| @end\_music | Gosu::Song | End Song |

Main.rb enumeration

|  |  |
| --- | --- |
| Value | Notes |
| Fate :count\_reached | Game lost when chicken cross border |
| Fate :hit\_by\_enemy | Game lost when collide with the chickens |
| Fate :off\_top | Game lost when move to close to the mothership |
| @scene :start | Start game |
| @scene :game | Begin playing |
| @scene :end | Game Ending |
| @counter | Count of chickens killed |

File2: bullet.rb

|  |  |  |
| --- | --- | --- |
| Field Name | Type | Notes |
| @window | Integer | Width and Height of the program |
| @x | Integer | X position of the bullet |
| @y | Integer | Y position of the bullet |
| @direction | Integer | Direction of the bullet |
| @image | Gosu::Image | Image of the bullet |

bullet.rb enumeration

|  |  |
| --- | --- |
| Value | Notes |
| Speed | Speed = 5 |
| Radius | Radius = 3 |

File3: enemy.rb

|  |  |  |
| --- | --- | --- |
| Field Name | Type | Notes |
| @window | Integer | Width and Height of the program |
| @x | Integer | X position of the chicken |
| @y | Integer | Y position of the chicken |
| @image | Gosu::Image | Image of the chicken |

enemy.rb enumeration

|  |  |
| --- | --- |
| Value | Notes |
| Speed | Speed = 2 |
| Radius | Radius = 20 |

File4: player.rb

|  |  |  |
| --- | --- | --- |
| Field Name | Type | Notes |
| @window | Integer | Width and Height of the program |
| @x | Integer | X position of the player |
| @y | Integer | Y position of the player |
| @image | Gosu::Image | Image of the player |
| @angle | Integer | Angle of the ship |
| @velocity\_x | Integer | Velocity of the ship |
| @velocity\_y | Integer | Velocity of the ship |

Player.rb enumeration

|  |  |
| --- | --- |
| Value | Notes |
| Rotation\_speed | Rotation speed = 3 |
| Acceleration | Acceleration = 1 |
| Fiction | Fiction = 0.9 |
| @radius | Radius = 20 |

File5: explosion.rb

|  |  |  |
| --- | --- | --- |
| Field Name | Type | Notes |
| @window | Integer | Width and Height of the program |
| @x | Integer | X position of the explosion |
| @y | Integer | Y position of the explosion |
| @image | Gosu::Image | Image of the explosion |

Explosion.rb enumeration

|  |  |
| --- | --- |
| Value | Notes |
| @radius | Radius = 30 |
| @finished | Finished = false |

# Overview of Program Structure

Moving functions:

1. Turn\_right

* Move the position of the ship to the right

1. Turn\_left

* Move the position of the ship to the left.

1. Accelerate

* Move the ship forward.

1. Decelerate

* Move the ship backward.

Gameplay functions:

1. Stage of the game (@scene):

* :start : start the game
* :game: begin the invasion
* :end: end the game

1. Spawn the player
2. Spawn the chickens
3. Spawn the bullet
4. Spawn the explosion whenever the bullet hit the chickens.
5. Count the number of killed chickens.
6. Lose the game when the chicken collides with the ship.
7. Lose the game if the chickens cross the border.
8. Game losing reason(fate):

* :count\_reached: The game is lost if the chicken cross the border
* :hit\_by\_enemy: The game is lost if the ship collide with the chicken
* :off\_top: The game is lost if the ship move to close to the mothership

1. After finish the game a scene will pop up to ask the player whether they want to play again or quit
2. A preformatted credit will run if the game is finished.

Structure chart:

Diagram

Description automatically generated