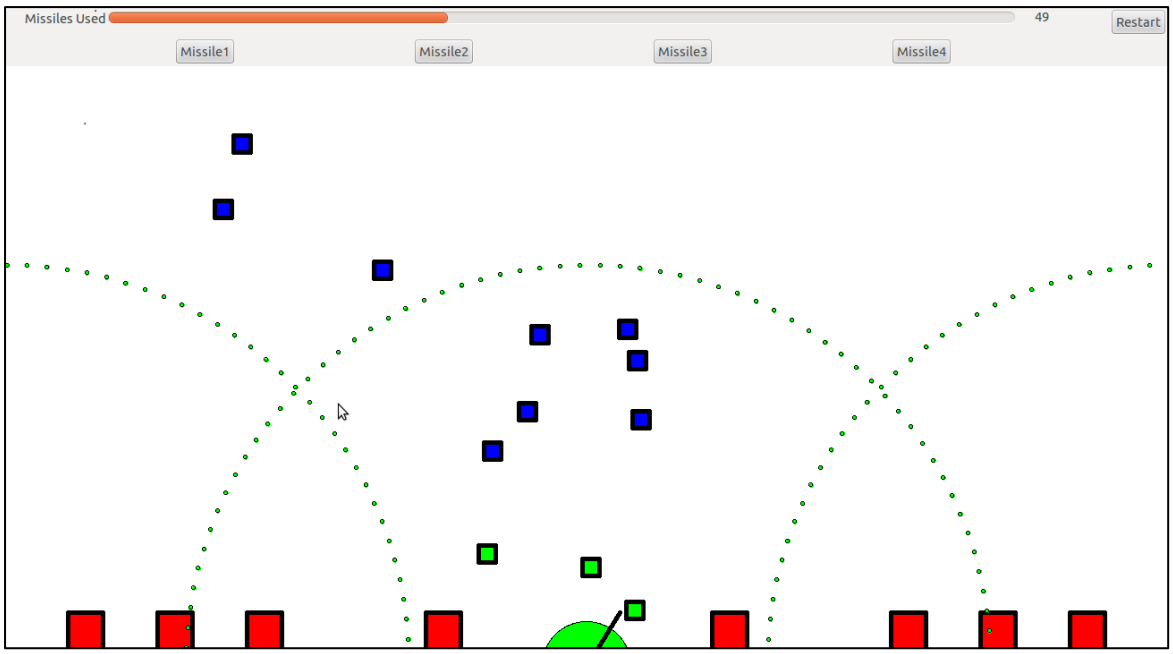
MISSILE COMMAND

Team Members

- Sahil Jindal 110020043
- Abhishek Gupta 110040067
- Mridul Ravi Jain 110040083

Brief Description:

This is a GUI based game, where the user has to completely destroy an enemy city using a fixed number of missiles he has at his disposal. On the other hand, there is an anti–missile gun which is controlled by the computer. The anti-missile gun calculates the path of the incoming missile (when it comes within the range of a radar) and releases an anti missile to destroy it.



Design of program: The game had various features like missiles, anti- missiles, anti – missile guns, buildings etc. This suggested use of different classes for each of them.

A. **Missile% :** It has various properties such as position of launch pad chosen, current position, time of launch , velocities in x and y directions and boolean fields which tell whether it is in radar's range and whether some anti- missile has been launched to destroy it. It also has an update member function which takes the current time as parameter and updates the current position of the missile object.

B. **Anti-missile%**: It has fields for position of anti missile gun from which it is launched , current position of anti missile, velocities in x and y directions, launch time and the missile object which it aims to destroy. This also has a similar update member function.

C. **Anti-missile-gun%:** It has fields for x and y coordinates of the gun, current angle it makes with positive x axis , final angle to which it must rotate before launching an anti missile , boolean member which determines whether the gun is free or not . It also contains a field for the missile object it is currently intending to destroy and fields for radius and rotation speed of the gun.

   There is an update member function which updates current angle as required and launches the anti missile if current angle is close to the final angle.

D. **Building%:** Since we have modelled the building as a rectangle, we have fields for position of left upper point of the rectangle, and for height and width.

How the design works :

- We maintain different lists for missiles, anti missiles and buildings which are "**active**" i.e they are still in consideration (have not been destroyed or gone out of frame).
- After each time increment, we update positions of all "active" objects in our simulation.
- We then check if any building has been destroyed by an incoming missile or if a missile has collided with an anti missile. In such a case, these objects cease to exist in the game.
- Finally, a function is called which tries to "**associate**" anti missiles with all missiles (within the radar's range) for which an anti missile has not been fired as yet.

Sample input - output:

- There are *10 buildings* and *an anti missile gun* which are placed at the bottom. Initially, the "cannon" of the gun points vertically upwards.
- The dotted semi circular regions represent the range of the *3 radars* of the city.
- *4 launch pads* for the missiles which are at the top of the screen. To launch a missile, the user needs to select a launch pad by clicking on it and then click anywhere on the screen to give direction with respect to the selected launch pad.
- You win the game (a dialog box appears) if all the buildings are destroyed without using up all the missiles in your arsenal.
- You lose the game if all the missiles are used up (but the city is not destroyed completely).
- You can restart the game any time you want to, by simply clicking on the '*Restart*' button.
- You can exit the game by clicking on the '*Exit*' button on the main menu page. On this page, you can also *select the level of difficulty*.

Limitations and bugs in our program:

- On pressing the Exit button, the *process is not killed*. But the windows are closed and the program can be run again without any problem.

- Due to time constraints, we implemented *only one anti-missile gun*. But it can be extended to include multiple guns which coordinate with each other to destroy missile closest to it.
- We have *not considered gravity* or any such factors. Hence the motion is strictly linear.
- The *anti missile can pass through the buildings*, even though actually they should also destroy the buildings!

Interesting features of the game/simulation:

- **GUI:** We have made a neat and tidy interface for the game, where the user can easily start, restart, exit, choose level of difficulty etc.
- The (semicircular) anti missile gun can't change the direction in which it is pointing instantaneously, i.e it takes time to rotate. This feature makes the game more realistic and improves the user's chances of winning.
- An anti missile is not launched if the missile under consideration will not collide with any of the buildings.

Interesting coding features :

- Extensive use of higher order functions like **map** and **foldr**.
- Definition and use of a **macro** describing the "**for**" loop.
- Use of "**racket/gui**" package for all the graphics. We implemented various objects like *buttons, dialog boxes, message boxes, choice menus, gauge* etc in our game.
- Instead of writing same code again, we defined a single function that would be called .

  For example, we have a "**set-to-initial-state**" function which sets all parameters to their initial values. This is used when game is restarted, or when the user wins/ loses and opts for the "play again" option.

Abstractions:

- The missiles and anti missiles have been *modelled as rectangles* moving in a straight line at const velocity.
- The missile and anti- missile need not meet/ collide "exactly", it is enough for the anti missile to be "close" to the missile.