SPACE INVADERS GAME

PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENT FOR THEAWARD OF THE DEGREE OF

BACHELOR OF TECHNOLOGY

(COMPUTER SCIENCE AND ENGINEERING)

SUBMITTED BY
**PRATHAM KUMAR
GUPTA
(2003490100022)**

15 SEPTEMBER 2022

Dr. APJ Abdul Kalam Technical University, Uttar Pradesh

LUCKNOW, INDIA

TABLE OF CONTENTS:-

OBJECTIVE:-

The aim is to destroy the rows of aliens as they move horizontally across the screen ,movingfaster and faster as you pass levels. Once you destroy one wave of aliens, get ready to face another faster and more difficult wave of aliens

If you allow any space invaders to reach the bottom of the screen or enemies bullets consumeyour lives , the alien invasion has been successful and the games ends.

- To control a spaceship and shoot down the enemy aliens to score points.
- To avoid colliding with the enemy aliens and their bullets.
- To progress through each level by destroying all enemy aliens.
- To use power-ups and bonuses to gain an advantage over the enemy aliens.
- To achieve a high score by achieving the objectives of each level

INTRODUCTION:-

Space Invaders(SI) origins date back to 1978, where the videogames were quite simple and there wasn’t the varietythat exists nowadays in video games, and so almost every game used to had a huge success. SI was originallymanufactured and sold by Taito in Japan and it is an excelent retro arcade video game. The objective in the gamewas to control a sort of space aircraft and defeat the aliens by shooting them with a laser and destroy as many as possible and then that performance was converted to points which would be then in an high score table. In its firstversions Space Invaders was really simple in a two dimensional environment game where the player moves his”spacecraft” to the left and right shooting the aliens. Space Invaders (1978) The reason for choosing the rebirth ofSpace Invaders [1] relates to the fact that this was a great game on his era and deserves recognition. This recognition process will be done by rethinking the core- ideology of the game and by adding some new features that will improvethe game dynamics.

Space Invaders is a stationary shooter in which the player fires at aliens overhead by moving a laser weapon horizontally across the bottom of the screen. As a group, the aliens travel left and right, shifting downward as they approach the screen’s edge. The objective is to shoot all of the aliens to death. The game finishes quickly if the invaders reach the bottom of the screen while the player has three lives.

SOFTWARE AND HARDWARE REQUIREMENTS:-

SOFTWARE REQUIREMENT

- Python
- Pygame
- OS-Windows
- Pycharm

HARDWARE REQUIREMENT

- with 8gb ram
- 512gb SSD
- Intel core i5 10th Generation
- Intel UHD Graphics

LITERATURE REVIEW :-

Space Invaders is one of the most iconic classic games in existence. It's up there with Tetris, Pong and Pac-Man as a quintessential game that has suffused the world's consciousness. If you asked around my high-school class you'd struggle to find a student without some exposure to Pac-Man through the Google doodle from about ten years ago but comparatively few students who have played the original arcade game. Similarly, I've definitely played some version of Space Invaders in my life but I couldn't tell you what version and I'm certain I've never played the original. This puts me in a very unusual position in this review of Space Invaders Invincible Collection where I'm covering entries in a franchise that I'm hyperaware of which were released well before I was born and before video game reviews were an established thing. But I'm nothing if not adaptable so I'm going to pull up my tube socks, put on some prog rock.

RESEACH PAPER 01:-

Department of Computer Science and Engineering, BNM Institute of Technology, Karnataka, India(Volume: 05 Issue: 06 | June -2018).

Abstract –

This project describes the creation of a web application using a space environment design to implement a game using virtual reality, with the use of a VR headset consisting of goggles that are used to create a virtual screen. The developed game enables users to exercise better degree of user control, thus providing greater user convenience. It allows them to envision the abstract environment clearly, thus enhancing its performance. The success of this game involved the usage of a number of concepts, over an a-frames platform. The developed game enables users to exercise better degree of user control, thus providing greater user convenience. It allows them to envision the abstract environment clearly, thus enhancing its performance.

1. INTRODUCTION :-

With the advancements in science, came the synchronous advancements in technology. Technology has brought about many changes in the field of medicine, research and agriculture. The field of gaming has been elevated with the concept of virtual reality. This concept brings together user interface and simulations of a virtual environment allowing gamers to witness realistic images, sounds and sensations to visualize their presence in a made-up environment. This project describes the creation of the same using a space environment design to implement a game using virtual reality, with the use of a VR headset consisting of goggles that are used to create a virtual screen. This technology constitutes the use of a virtual reality box, in order to produce sounds and images that allow the user to truly picture himself in the holographic projectionThe hardware components used in lieu with the software enhances the vision of the user, granting him access to view and move around the artificial environment, whilst interacting with the items present in the same. The concepts pertaining to tele-existence and tele-presence are incorporated into the environment model created, to build a complete and apt design. By manipulating the various keyboard and mouse controls, one can move in a given direction within the created space. Recent technological advancements pertaining to the building of wrap-around display screens and haptic devices allows users to feel the presence of display images. To create a true spatial environment, it is necessary to merge the content of immersion. It constitutes the perception of user of being physically present within the space. The images and sounds within the same build up this perception, acting as stimuli that allow users to remain engrossed in the environment. The gamification process adds a complication to the game that is build, integrating various game design concepts, thereby increasing difficulty of the game. The user interaction and learning is enhanced with the introduction of this concept. Study and analysis relating to psychology and behavior of human nature is undertaken to provide better learning outcomes. Before building the same, it is necessary to first design the modules and collect data used in the creation of the model. Deciding the various inputs and outputs, thus declaring the interface is the initial step in the planning process. Flowcharts and algorithms are used to clarify the flow of control within the code. Optimization techniques are operated on the code to reduce memory consumption. The main purpose of designing an educational game pertains toward the need to incorporate learning in the field of entertainment. This is done in order to capture the attention of the user. The basic need as a developer is to ensure that one captures the essence of learning within the gaming environment. With the ever-increasing growth in technology, there is a simultaneous growth in the education domain. Developing the same enhances the efficiency of building an effective and classic base for learning. Building a game that provided education through gaming is now a popular strategy to help improve the concentration of the user. Learning can be made easy through the concepts used to develop the same.

2. SYSTEM DESIGN

The system built brings together the elements of HTML, JavaScript, JSON and React while incorporating the features of A-frames and it is built over the concept of Virtual Reality. The system is implemented in Redux to work as an Android and IOS application. The educational game prototype is designed such that the system should: (1) run on both personal computers and phones, (2) compatible with all VR headset, (3) playable in non-Virtual Reality devices, (4) detect movement, (5) give reaction to user input and (6) shoot a bullet when the button is triggered. Along with the mentioned functional requirements, the system should possess the following quality attributes (1) contain concepts of space, (2)provide simple user interface, (3) give quick reactions to user inputs and run smoothly, (4) provide good user experience and (5) have striking graphics. The game consists of eight levels, which is equivalent to the number of planets in the solar system. The game is education-based, where the user gets to know about the planets in the solar system in detail and it enables one to test his/her ability to answer the quiz in an efficient manner. Initially, the main screen is displayed, which consists of all the planets, along with the sun. Then the user has to select mercury as his first level and clear it by combatting the aliens. Once all the aliens in a level are killed, before the time runs out, the user will be able to attempt the quiz, else the user loses. Once the level is cleared the user will be taken to the next level. And in the similar way the user has to invade the space by capturing all the planets.LI.

3. DATA FLOW DIAGRAMS

The figure below shows a context Data Flow Diagram or Level 0 Data Flow Diagram drawn for the system. It is an overview of the whole system. In Level 0 DFD, the game and gamers relationship is shown, and the gamer interacts with application using gamer inputs. The gamer ends the application after completing the game and the quiz.

The Level 1 Data Flow Diagram is the decomposition of the context Data Flow Diagram. It involves the breakdown of the main process into sub processes. In Level 1 DFD, the operation of the game is explained. The gamer interacts with the game using game controllers. Data flow Di

4. RESULTS

The snapshots show the working of the system. Initially, when the user launches the application, the user sees the initial screen which consists of the solar system containing all the planets, sun and the celestial bodies. It is in the initial screen where the user chooses the planet or level to be played. The difficulty increases with the distance of the planet from the sun. Fig -4: The main screen displaying the Solar System After the user chooses a particular planet, the information of the planet is displayed which helps the user gain knowledge about the planet. The information screen is visible for 100 seconds. Fig -5: Displaying the information for the planet In the next screen, the user encounters the aliens and has to combat all the aliens in the given time in order to play the quiz, else the player has to restart the game by choosing the level again. The screen consists of a timer and score which varies with each planet.

RESEACH PAPER 02:-

Abstract: This paper documents the design and development of a 2D computer game Duel Space Invaders(July 2020).

1. INTRODUCTION

Computer programming is about creating abstractions. The abstract objects are implemented in order to realise useful programs which solve problems, improve quality of life and design games to promote creativity and idea generation. This report documents the design, implementation, testing and analysis of the Dual Space Invaders arcade game. Space Invaders is an arcade that was released in 1978 by Taito Corporation and it is considered to be one of the first arcade games created [1]. Dual Space Invaders is a version of Space Invaders which has two players. The Dual Invaders game offers the user two player objects which are at the top and bottom of the screen and they can both be used in the ”single player” & ”dual player” game modes. The Single player mode allows the user to control the player object at the bottom of the screen while the other player object mirrors the movement and shooting demeanour of the first player. Furthermore, the dual game playing mode enables the functionality of both game objects to move and shoot independently provided that there are two different users to regulate each game object. A good way to boost the user’s self-confidence and encourage one to continuously play the game is ensuring the user is always engaged with the game, this is achieved by preserving the high score of every game such that in the next game, the user can try to exceed the current high score. It has to be noted that the two game playing objects have a common score which progressively adds to the current high score. The game ending state is triggered when one of the following event occurs: all the aliens are destroyed by the player object, either of the player laser cannons are destroyed by shooting each other or the aliens shots and when the aliens seize the player’s territory i.e. entering the top or bottom row. The scope of the report

2. SOFTWARE DESIGN

Software Architecture The design of the game architecture is implemented based on the concept of Separation of Concerns, this concept makes it convenient to develop, maintain and test different functions of the design. This concept is a major feature of the Layered Architectural Pattern which also serves as the building block of the game software architecture. The benefits of using the Layered Architectural Pattern[2] is that it allows for isolation of layers in order to make it convenient to maintain and test each layer. The isolation of layers ensures that there isn’t a cause-effect i.e. changing one layer which ultimately affects other layers, this approach ensures that the presentation layer doesn’t have knowledge of the logic layer, as the logic layer has limited or small knowledge of the presentation layer. Figure 1 denotes a high level representation of interactions between game layers.

3. THE DATA LAYER

The data layer serves as the second phase of the Layered Architectural Pattern of the Dual Space Invader Game design. This layer’s main purpose is to store the information of the various game objects and it provides information to the logic layer to be processed, aggregated and interpreted. The presentation layer also receives information such as coordinates and life state of the objects from the data layer in order to draw the sprites.

4.1 GameEntity

The GameEntity class is the backbone of the data layer architecture. This class plays a vital role in the creation of game objects as it initialises the game object's coordinates and ensures that they are valid i.e. the coordinates are within the boundary of the Render Window. Invalid coordinates are handled by InvalidEntityPositions exception. The Game Entity ensures that every created game object has a boolean variable life = true and the member function giveEntityLife() is called to realize the life state of the game object, whilst the destroyEntity() member function destroys an object's life by ensuring that the boolean life = false. The game is implemented such that the LaserCanon object has a limited lifespan, therefore the GameEntity class initialises and returns the number of lives for every created LaserCanon object. The GameEntity class is a parent class to other classes of the data layer which inherit its properties and methodology, refer for the inheritance hierarchy of the Game Entity class. Inheritance diagram of Game Entity

4.2 MovingEntity

This class is a child class of the GameEntity class and it inherits various properties which aid in fulfilling its primary objective that is to regulate the movement of the game objects. The constructor of this class initialises the speed of the player and the getEntitySpeed() member function returns the speed of the LaserCanon object.

4.3 Ammunition

The Ammunition class is a child class of the GameEntity class and it inherits the capability to set life = true when a game object shoots a laser, this is achieved by calling the member function giveEntityLife().

4.4 LaserCanon

As seen in the LaserCanon class is derived from the MovingEntity class in order to allow a player to move in the player's desired direction. This class is also responsible for continuously updating the score and high score of the player and the game, respectively. Furthermore, this class throws FileCannotBeOpened exception if a text file which contains the score and high score cannot be opened. It has to be noted, that the InvalidLaserCanonCoordinates exception is thrown provided that a LaserCanon object is to be created at coordinates which are not recognised by the design of the game.

4.5 LaserCanonLife

This class is responsible for keeping data on the number of lives a player has during the gameplay. Hence, the LaserCanonLife class is a child class of the GameEntity class. The Laser Canon has a total lifespan of 3 and a player life decreases when an Alien shoots the player and if the two players collide or when they shoot at each other. This class This subject of collision handling and detection is further elaborated on Section 5.3.

4.6 Laser

Although Laser Canons are able to shoot lasers, it is important that game objects such as the Aliens and are capable of shooting lasers in order to make the game more challenging. The Laser class ensures that Laser objects of the Aliens or the LaserCanon objects are able to move vertically to the direction the object is facing. This class is a child class of the MovingEntity class as it is imperative that it inherits the movement functionality. Creating a Laser object for an Alien class object requires that the created object has the same coordinates as the alien, therefore, this class throws a InvalidAlienLaserCoordinates exception if this condition is violated. Similarly, a

InvalidLaserCanonCoordinates exception is executed should a Laser object of the LaserCanon be initially created outside of the bottom or top boundaries of screen which are specified initial positions of the laser canons.

4.7 Alien

There exists a ”has-a” relationship between the Laser class and Alien class, which means that destroying an alien also destroys the laser associated with the specific alien. The game consists of two sets of Aliens; the first set of three rows of aliens move upwards while the second set moves downwards. The Alien class ensures that each set of aliens have the movement functionality, hence it is a subclass of the MovingEntity class. The class ensures that movement of the aliens is restricted to the boundaries of the screen, provided that the alien object is initialized with coordinates that are not out of the scope of the game window therefore an InvalidAlienCoordinates exception is responsible to avoid violation of this condition. An enumeration AlienColour provides colours for red, purple and green aliens.

4.8 LaserCanonShields

The game offers security to the laser canons by creating LaserCanonShields objects which can handle the impact of the laser shots from the alien. However, player shield objects of the LaserCanonShields are destroyed once they reach an unstable state. This class is derived from the GameEntity class and inherits the properties of the parent class by setting the life state of every LaserCanonShields object. A shield’s life state life = true upon creation and the member function giveEntityLife() is called to realize the life state of the game object, whilst the destroyEntity() member function destroys an object’s life .

5. THE LOGIC LAYER

The logic layer is the game engine which fuels the logic of the Dual Space Invader game. This layer manages the demeanour, interaction and the overall conversations between game objects. The Layered Architecture Pattern as indicated in Figure 1 ensures that the logic layer gathers information from the data layer in order to process and make decisions which produce a logical output.

5.1 GameLoop

The GameLoop class serves as the skeleton which the game engine is based on. This class is responsible for the creation of all game objects, maintaining game activities and the carrying out overall functionality of the game. In essence, the GameLoop class is the interaction of the presentation and data layer in the Layered Architectural Pattern design. Figure 4 in the Appendix indicates the Sequence of interactions between objects of GameLoop class. The GameLoop class allows for conversations between objects during gameplay which includes processing data from the data layer in order to process and make decisions which produce a logical output to the presentation layer.

5.2 GameUpdater

Another important aspect of the game logic is the ability to realize the decisions which are made by objects as a result of different user inputs and game activities. The GameUpdater class constantly updates the game as a consequence of events which occur during gameplay, therefore upon movement and collisions of the game objects this class updates the coordinates.

5.3 Collision Detector

The Collision Detector class’ primary purpose is to handle the collision of game sprites/objects. It is imperative that collisions are detected and handled as this denotes if a player has won or lost the game or if a player has lost a life. This class caters for collisions between the following game objects:

• Laser Canons: A player loses a life if two Laser Canons collide. The game ends and the player loses if there are no lives remaining.

• Laser Canon and Laser: A collision occurs if an Alien shoots the Laser Canon or if the Laser Canons shoot each other. In both instances, the player loses a life and if there are no lives remaining the game ends in a loss.

• Laser Canon and Laser Canon Shield: A Laser Canon object cannot move through the Laser Canon Shield if and only if the Laser Canon Shield object is not destroyed.

• Laser Canon Shield and Laser: Alien objects continuously shoot at the Laser Canons and the shields provide protection. Collisions between Laser Canon shields and Lasers are handled by this class provided that the shield is still present and has not been utterly destroyed by the lasers

Alien and Laser: In order for the player to win the game, all alien objects are to be destroyed by the Laser Canons regardless of the game playing mode. Aliens are destroyed by the Laser shot of the Laser Canon objects upon instruction of the

6. CONCLUSION

The design of the Dual Space Invader game based on the Layered Architectural Design is successfully implemented. The overall design successfully exceeds the minimum requirements as it satisfies basic functionality which includes five minor and two major feature enhancements. Furthermore, the design and implementation is such that the key programming principles which are the DRY principle, separation of concerns and code re-usability. Unit testing is performed in order to identify errors validate the functionality of the source code. An analysis of the overall design commends the strengths and exposes the weaknesses of the quality of the design, however, further improvements are to be made to the overall design in order to derive a robust and efficient design to improve overall functionality and quality of the design.

RESEACH PAPER 03: (D. Gries, L. Lee, S. Marschner, & W. White **Fall 2017)**

If you have never played Space Invaders before, there are a few versions online as flash games. The Pacxon version is the most authentic one that we have found. You should play this to get a good idea of the gameplay. Keep in mind that this version is a bit *too* authentic and potentially runs afoul of copyright issues.

One of the main challenges with this assignment is its scope is completely up to you. There is a bare minimum of functionality that you must implement. You must implement a complete, single wave of Space Invaders. But after that point, you are free (and encouraged) to add more interesting features to your game. The video to above shows our solution, which has several extra features such as sound and basic animation. You can even look at more advanced shooting games like *Galaxian* and *Galaga* for inspiration. You are permitted to do anything that you want, provided that the basic functionality is there.

Academic Integrity

We had to abandon the Breakout assignment because the number of Academic Integrity cases had gotten out of hand. It was a classic assignment used by both us and Stanford (though theirs was subtly different) and there were a lot of versions floating around on the web.

This is a brand new assignment, so the chances of you finding helpful code online are pretty slim (we looked). But you still may find some code out there. There is also a lot of code available for you to use in the lecture on GUI applications. The animation, arrows and subcontroller examples are particularly useful. You are permitted to copy whatever code you want from these (or other samples).

Copyrighted Material

Gameplay cannot be copyrighted. You can make a game that plays the same as another. Indeed, it There is another Academic Integrity issue with this assignment: copyrighted material. was *Space Invaders* itself that lost the court case (against *Galaxian* and *Galaga*) that established this fact. However, artwork in a game is copyrighted. So you should not attempt to use the original *Space Invaders* characters for your game.

While there is maybe an argument for fair use, since this is a class project, your instructor prefers that you avoid the copyright issue entirely. The *Space Invaders* are iconic. While they may not appear in video games these days, they are still sold on T-shirts and appear in bad Adam Sandler movies. Furthermore, the current rights holder is a little company called Square Enix, which is not afraid of lawsuits.

In general, you are only allowed to use copyrighted material if you have a license to do so. For example, many of the songs and sound effects in the NewGrounds library are available for you to use under an *Attribution License*. That means you are free to use it so long as you cite the source in your documentation (e.g. your header comments). This is okay. A license where you have to pay is not okay.

Organization and Scope

This is a long assignment, similar in length to Assignment 6. Once again the trick is to pace yourself. This can be finished by the end of classes, if you work a little bit every day (excluding the Thanksgiving break).

While there are no unit tests this time, you should be able to figure out if everything is working simply by playing the game. There are no tricky "restore everything to how it was" like with Turtles. Just get the game working.

Assignment Source Code

The first thing to do in this assignment is to download the zip file code.zip from this link. You should unzip it and *put the contents in a new directory*. As with the imager application, this assignment is organized a package with several files. In particular, this package file contains the following:

app.py
: This file contains the **controller** class Invaders. This is the controller that launches the application. It is one of three modules that you will modify for this assignment. While it is the primary controller class, you will note that it has no script code. For that, you will use the module __main__.py below.

wave.py
: This file contains the secondary **controller** class Wave. This class manages a single wave of aliens. It works as a subcontroller, just like the example subcontroller from class. It is another of the three modules that you will modify for this assignment, and the one that will require the most original code.

models.py
: This file contains the **model** classes Ship, Alien and Bolt. If you want to add other model classes (e.g. power-ups), then you should add those here as well. This is the last of the three files you should modify for this assignment.

consts.py
: This is a module filled with constants (global variables that should not ever change). It is used by app.py, wave.py, and models.py to ensure that these modules agree on certain important values. It also contains code for adjusting your alien count and speed. You should only modify this file if you are adding additional constants as part of your extended features.

__main__.py
: This module contains the application code for this assignment. It is the module you run from the command line to start the game. It works the same way that the imager application did in Assignment 6. *Do not modify this file*!

game2d
: This package contains the classes you will use to design you game. Technically, this was already installed as part of Cornell Extensions. However, we have made some important modifications for this assignment. *Under no circumstances should you ever modify this package*!

Sounds
: This folder is a list of sound effects that you may wish to use as part of your extensions. You are also free to add more if you wish; just put them in this folder. All sounds must be WAV files. While we have gotten MP3 to work on Windows, Python support for OS X is unreliable.

Fonts
: This folder is a collection of True Type Fonts, should you get tired of the default Kivy font. You can put whatever font you want in this folder, provided it is a .ttf

file. Other Font formats (such as .ttc, .otf, or .dfont) are not supported.

Images

This folder is a collection of images for the ship and aliens. The GImage and GSprite classes allow you to include these in your game. You may also want to include other images, such as a background; just remember to draw the background image *first*.

For the basic game, you will only modify the first three files (maybe four) listed above. The class Invaders is a subclass of the class GameApp. Your model classes should all be subclasses of GObject. As part of this assignment, you are expected to read the online documentation which describes how to use the basic classes.

Running the Application

This application is very similar to Assignment 6 in that it is organized as a package. To run the application, make sure all of the files are together in a folder and give the folder a name like `invaders`. To run the program, change the directory in your command shell to just *outside of the folder* *`invaders`* and type

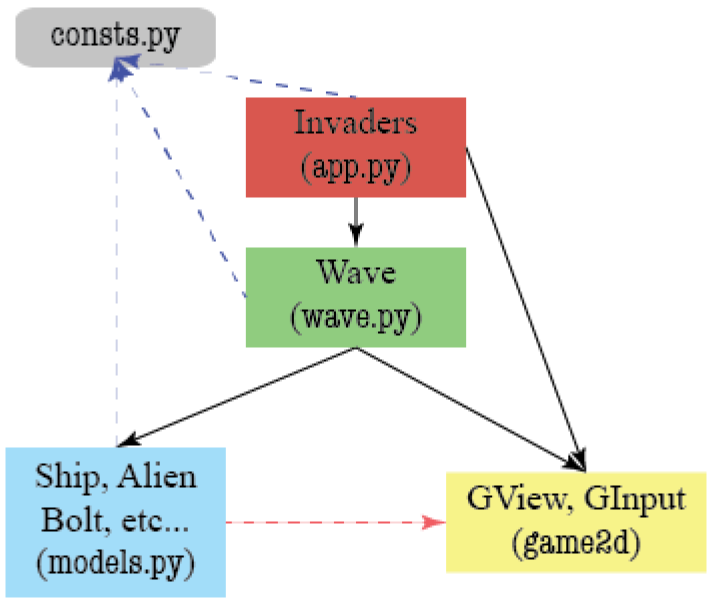
```
python invaders
```

In this case, Python will run the *entire folder*. What this really means is that it runs the script in `__main__.py`. This script imports each of the other modules in this folder to create a complex application.

Assignment Organization

This assignment follows the model-view-controller pattern discussed in class. The modules are clearly organized so that each holds models, the view, or a controller. The organization of these files is shown below. The arrows in this diagram mean "imports". So the Invaders controller imports the view and Wave subcontroller. The Wave controller imports the view and the models. This leads to an important separation of files. Invaders is never permitted to access anything in models.py and Wave is never permitted to access anything in app.py. This is an important rule that we will enforce while grading.

You will notice that the models module imports the view because it needs the parent class GObject to perform any drawing. In practice, we often like to separate the model and view to cut down on the number of arrows (less meetings between the various programmers). However, that would make this assignment a lot harder. Fortunately, the view does not import anything (and should not be modified). This means there are no cycles in this architecture (e.g. A imports B imports C imports A). Cyclical imports are very dangerous and you have to be careful with them in large applications. Avoiding cycles is one of the reasons we draw pictures like the one below.



Invaders

This controller does very little. All it does is keep track of the game state (e.g. whether or not the game is paused). Most of the time it just calls the methods of Wave, and Wave does all the work. However, if you need anything between games, like a paused message or a high score, this goes here.

Wave

This class does all the hard work. In addition to the initializer, it needs its own update and draw methods. This is a subcontroller, and you should use the subcontroller example from class as a template.

The most complex method will be the `update` and you will certainly violate the 30-line rule if you do not break it up into headers. For the basic game, this method will need to do the follow:

- Move the ship according to player input
- March the aliens across the screen
- Fire a laser bolt from either the ship or an alien
- Move any laser bolts across the screen
- Resolve any collisions with a laser bolt

In our code, each one of these is a separate helper. You should think about doing this in your code as well.

The Models

The models just keep track of data. Most of the time, models just have attributes, with getters and setters. Think `Image` and `ImageHistory` from the previous assignment. However, sometimes models have additional methods that perform complex computation on the data, like `increment` or `undo` in class `ImageHistory`.

The models in this assignment are the game objects on screen: the ship, any aliens, and any laser bolts. Of these three, it is more important to have a class for `Bolt`. `Bolt` needs an additional attribute for its velocity, and you need some extra methods to perform calculations with this velocity.

The classes `Ship` and `Alien` are subclasses of `GImage`. You should not need any new attributes for these two classes. However, you will want to write a `collides` method in each of these classes to detect collisions with a laser bolt.

Game State

One of the challenges with making an application like this is keeping track of the *game state*. In the description above, we can identity several distinct phases of the game:

- Before the game starts, and the alien wave has not started
- When the aliens are set up, but have not started to move
- While the game is ongoing, and the aliens are on the march
- While the game is paused (e.g. to show a message)
- While the game is creating a new ship to replace the old one
- After the game is over

Keeping these phases straight is an important part of implementing the game. You need this information to implement `update` in `Invaders` correctly. For example, whenever the game is ongoing, the method `update` should instruct the `Wave` object to move the ship. However, if the game has just started, there is no `Wave` object yet, and the method `update` should create one.

For your convenience, we have provided you with constants for six states:

- `STATE_INACTIVE`, before a a wave has started
- `STATE_NEWWAVE`, when it is time to create a new wave of aliens
- `STATE_ACTIVE`, when the game is ongoing and the aliens are marching
- `STATE_PAUSED`, when the game is paused to display a message
- `STATE_CONTINUE`, when the player is waiting for a new ship
- `STATE_COMPLETE`, when the game is over

Completing the Assignment

Before submitting anything, test your program to see that it works. Play for a while and make sure that as many parts of it as you can check are working. Remember to check both of the lose conditions, not just the loss of three lives.

When you are done, reread the specifications of all your methods and functions (including those we stubbed in for you), and be sure that your specifications are clear and that your functions follow their specifications. If you implemented extensions, make sure your documentation makes it very clear what your extensions are.

As part of this assignment, we expect you to follow our style guidelines:

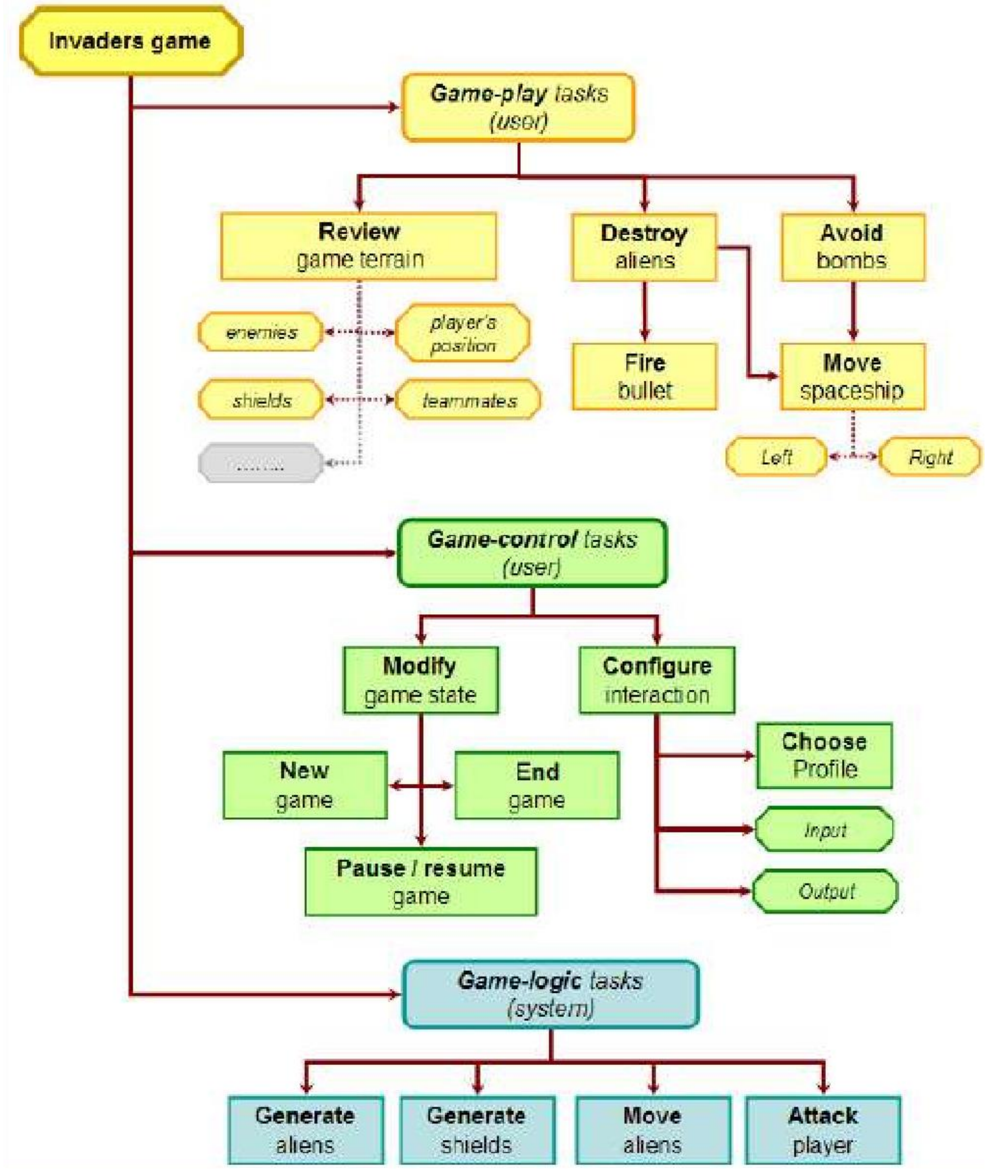
1. There are no tabs in the file, only spaces
2. Classes are separated from each other by two blank lines
3. Methods are separated from each other by a single blank line

4. Class contents are ordered as follows: getters/setters, initializer, non-hidden methods, hidden methods
5. Lines are short enough that horizontal scrolling is not necessary (about 80 chars is long enough)
6. The specifications for all of the methods and properties are complete
7. Specifications are immediately after the method header and indented
8. **No method is more than 30 lines long, not including the specification**

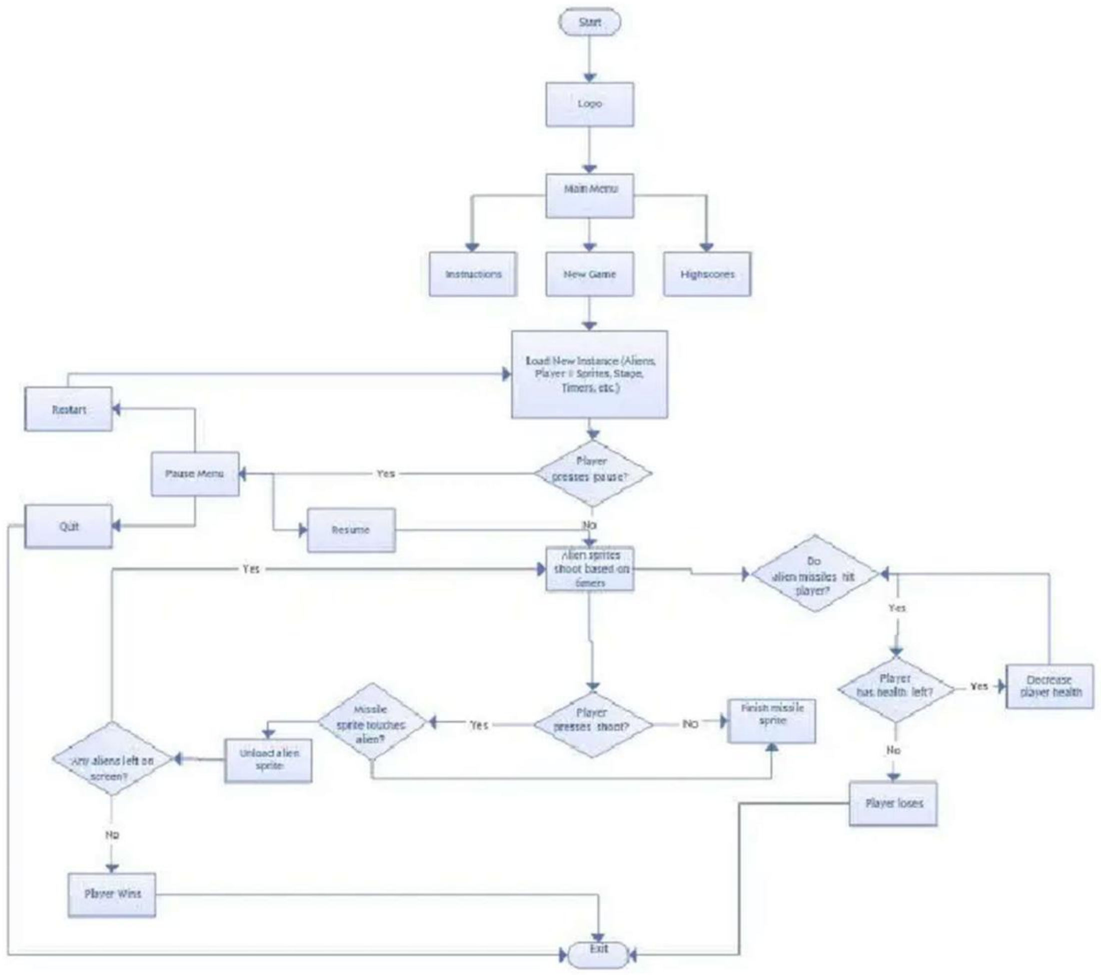
Survey

One last time, we need you to do a survey. The survey should be done individually (even if you worked in a group). As always, the survey will ask about things such as how long you spent on the assignment and your impression of the difficulty. Please try to complete the survey within a day of turning in this assignment. Remember that participation in surveys comprises 1% of your final grade.

ER DIAGRAM:-


Invaders game
Game-play tasks (user)
Review game terrain
enemies
player's position
shields
teammates
....
Destroy aliens
Fire bullet
Avoid bombs
Move spaceship
Left
Right
Game-control tasks (user)
Modify game state
New game
End game
Pause / resume game
Configure interaction
Choose Profile
Input
Output
Game-logic tasks (system)
Generate aliens
Generate shields
Move aliens
Attack player

FLOWCHART:-


Start
Logo
Main Menu
Instructions
New Game
Highscores
Load New Instance (Aliens, Player + Sprites, Stage, Timers, etc.)
Restart
Pause Menu
Yes
Player presses pause?
Quit
Resume
No
Alien sprites shoot based on timers
Yes
Do alien missiles hit player?
Yes
Player has health left?
Yes
Decrease player health
No
Player loses
Missile sprite touches alien?
Yes
Player presses shoot?
No
Finish missile sprite
Any aliens left on screen?
Unload alien sprite
No
Player Wins
Exit

TECHNOLOGY USED:-

PYTHON:-

Python is a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation.
Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library
Guido van Rossum began working on Python in the late 1980s as a successor to the ABC programming language and first released it in 1991 as Python 0.9.0. Python 2.0 was released in 2000

PYCHARM:-

PyCharm is a dedicated Python Integrated Development Environment (IDE) providing a widerange of essential tools for Python developers, tightly integrated to create a convenient environment for productive Python, web, and data science development. PyCharm is a
cross-platform IDE that works on Windows, macOS, and Linux.

- PyCharm is the most popular IDE used for Python scripting language.
- This chapter will give you an introduction toPyCharm and explains its features.
- PyCharm offers some of the best features to its users and developers in thefollowing aspects:
- Code completion and inspection
- Advanced debugging
- Support for web programming and frameworks such as Django and Flask

Features of PyCharm:

Besides, a developer will find PyCharm comfortable to work with because of the features mentioned below:

Code Completion:

SQLAlchemy as Debugger:

You can set a breakpoint, pause in the debugger and can see the SQL representation of the user expression for SQLLanguage code.
PyCharm enables smoother code completion whether it is for built in or for an external package.

Git Visualizaion in Editor :

When coding in Python, queries are normal for a developer. You can check the last commit easily in PyCharm as it has the blue sections that can define the difference between the last commit and the current one

.

Code Coverage in Editor:

You can run .py files outside PyCharm Editor as well marking it as code coverage details elsewhere in the projecttree, in the summary section etc**.**

Package Management:

All the installed packages are displayed with proper visual representation. This includes list of installed packagesand the ability to search and add new packages

User Interface of PyCharm Editor:-

1. The user interface of PyCharm editor is shown in the screenshot given below. Observe that the editor includes various features to create a new project or import from an existing project

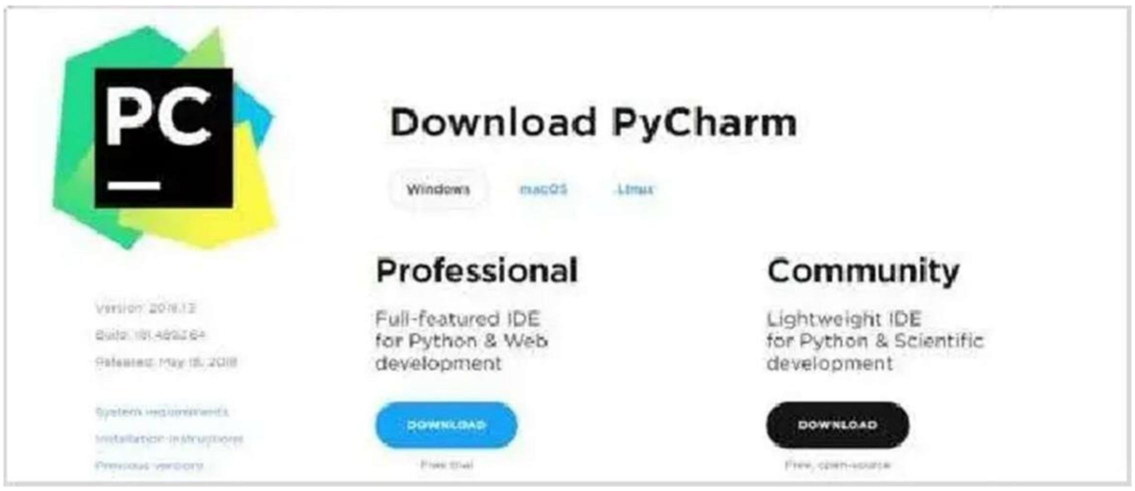
2. PyCharm – Installation

In this chapter, you will learn in detail about the installation process of PyCharm on your local computer.

Steps Involved :
You will have to follow the steps given below to install PyCharm on your system. These steps show the installation procedure starting from downloading the PyCharm package from its official website to creating a new project.

STEP 1:

Download the required package or executable from the official website of PyCharmhttps://www.jetbrains.com/pycharm/download/#section=windows. Here you will observe two versions of package for Windows as shown in the screenshot given below

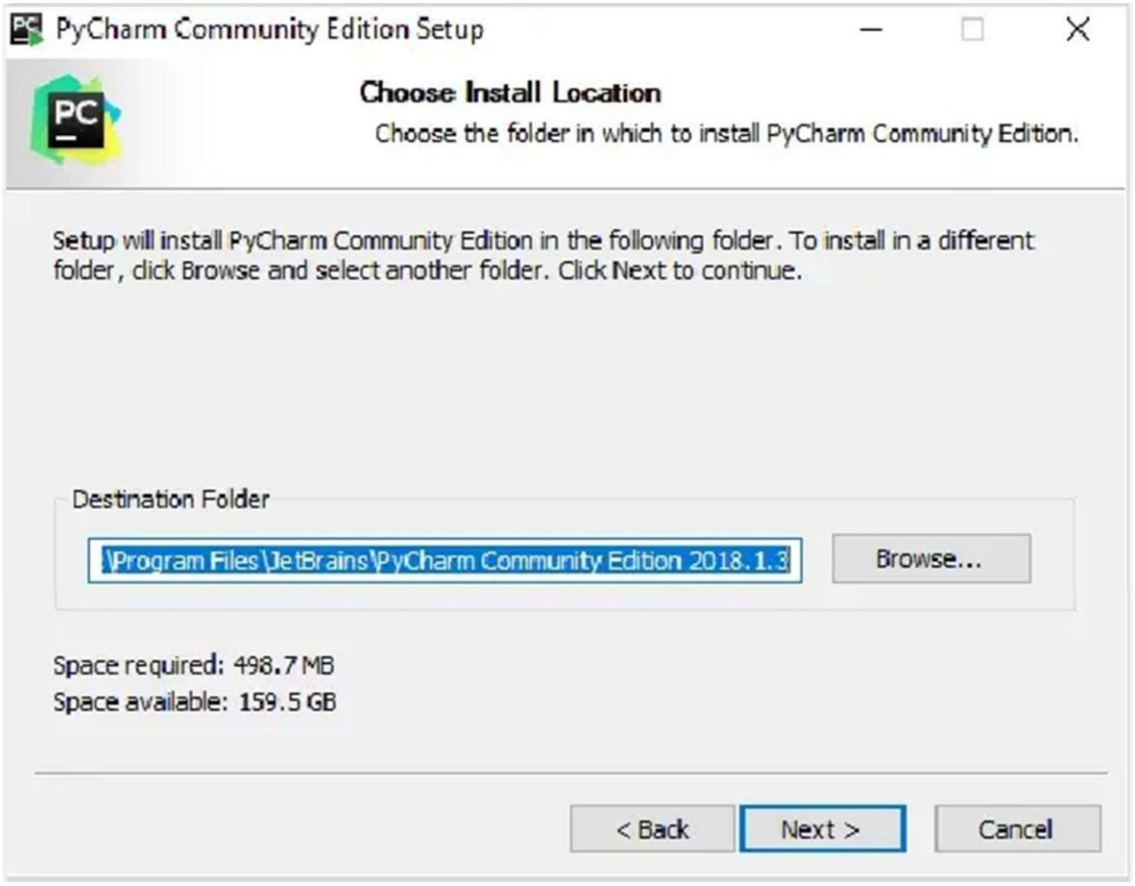


Note that the professional package involves all the advanced features and comes with free trial for few days and the user has to buy a licensed key for activation beyond the trial period.
Community package is for free and can be downloaded and installed as and when required. It includes all the basic features needed for installation. Note that we will continue with community package throughout this tutorial.

STEP 2:

Download the community package (executable file) onto your system and mention a destination folder as shown below

.


PyCharm Community Edition Setup

Choose Install Location

Choose the folder in which to install PyCharm Community Edition.

Setup will install PyCharm Community Edition in the following folder. To install in a different folder, click Browse and select another folder. Click Next to continue.

Destination Folder

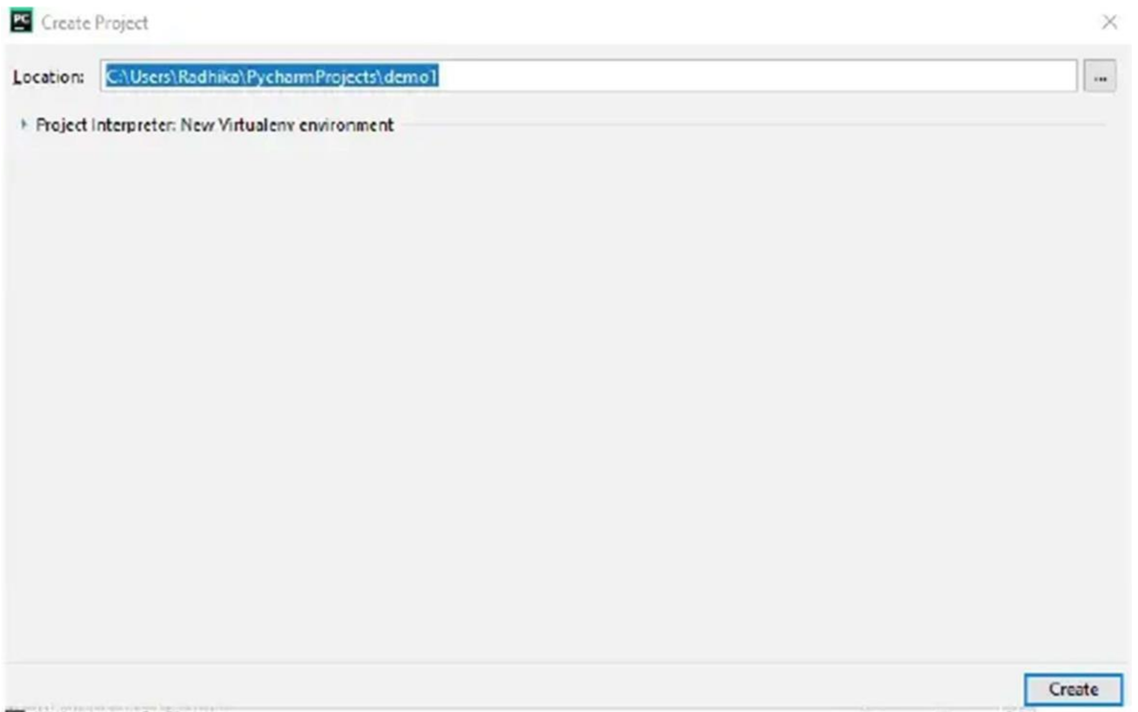
\Program Files\JetBrains\PyCharm Community Edition 2018.1.3

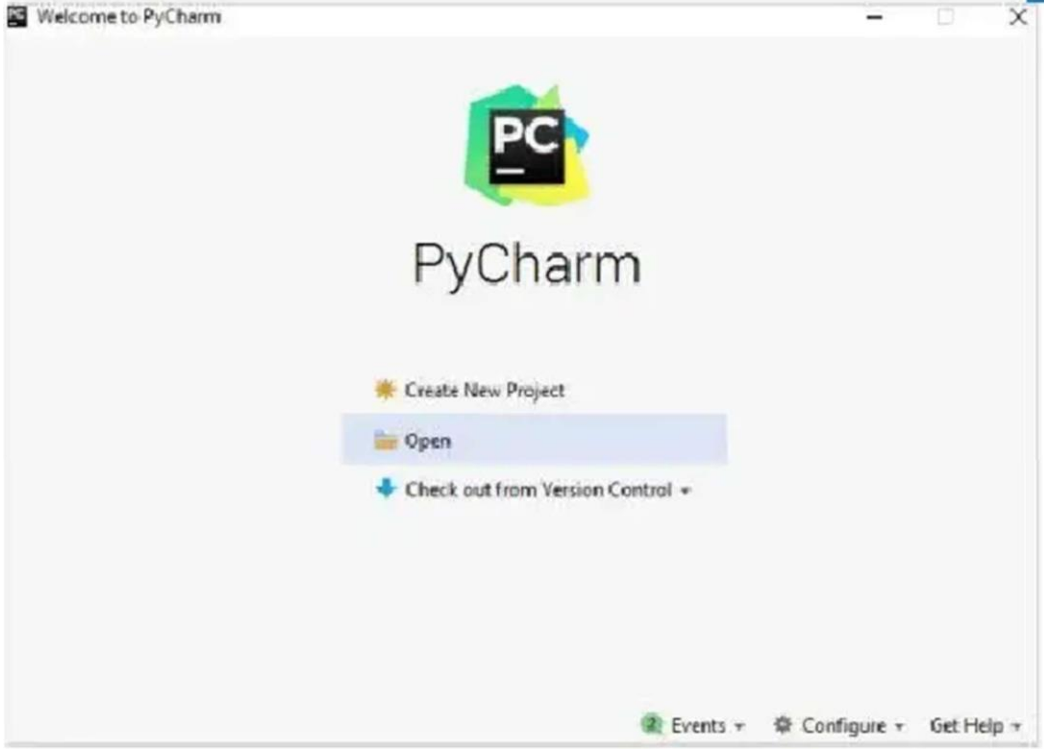
Space required: 498.7 MB
Space available: 159.5 GB

STEP 3:

Now, begin the installation procedure similar to any other software package

STEP 4:
Once the installation is successful, PyCharm asks you to import settings of the existing package if any


Create Project
Location: C:\Users\Radhika\PycharmProjects\demo1
Project Interpreter: New Virtualenv environment
Create

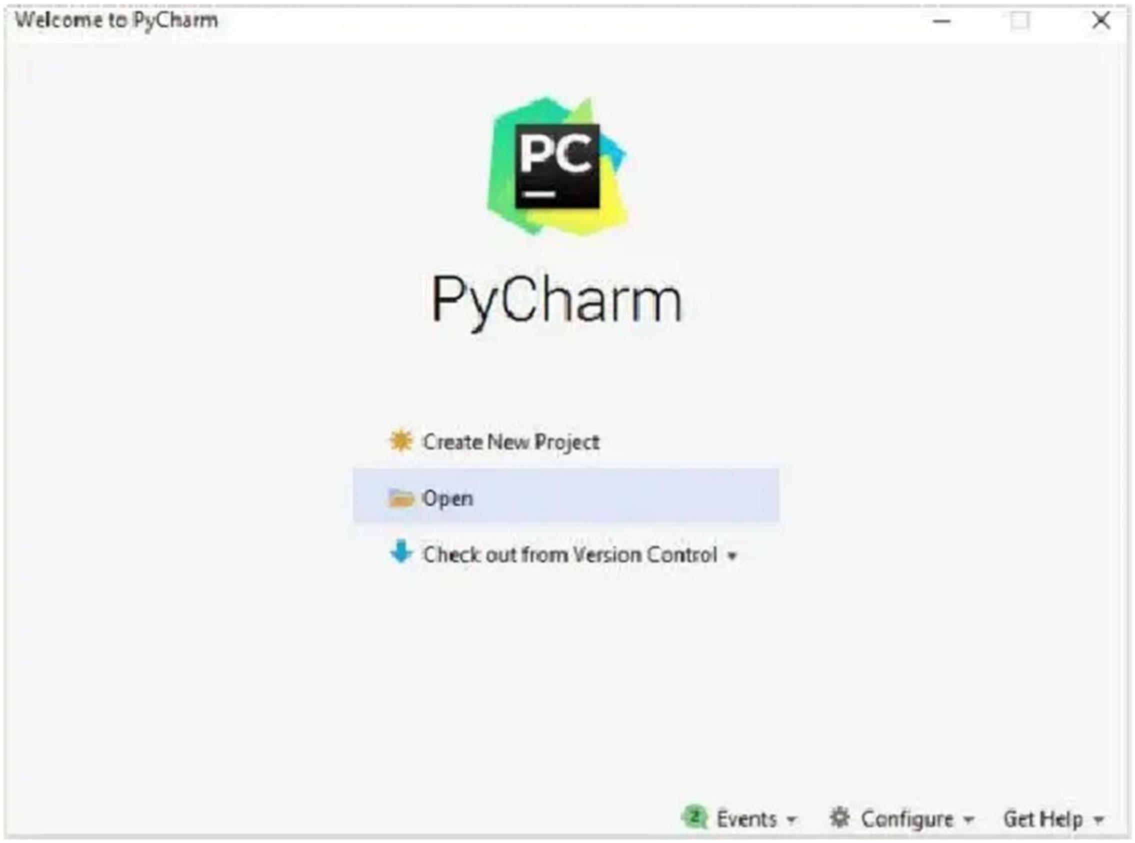


This helps in creating a new project of Python where you can work from the scratch. Note that unlikeother IDEs, PyCharm only focusses on working with projects of Python scripting language.

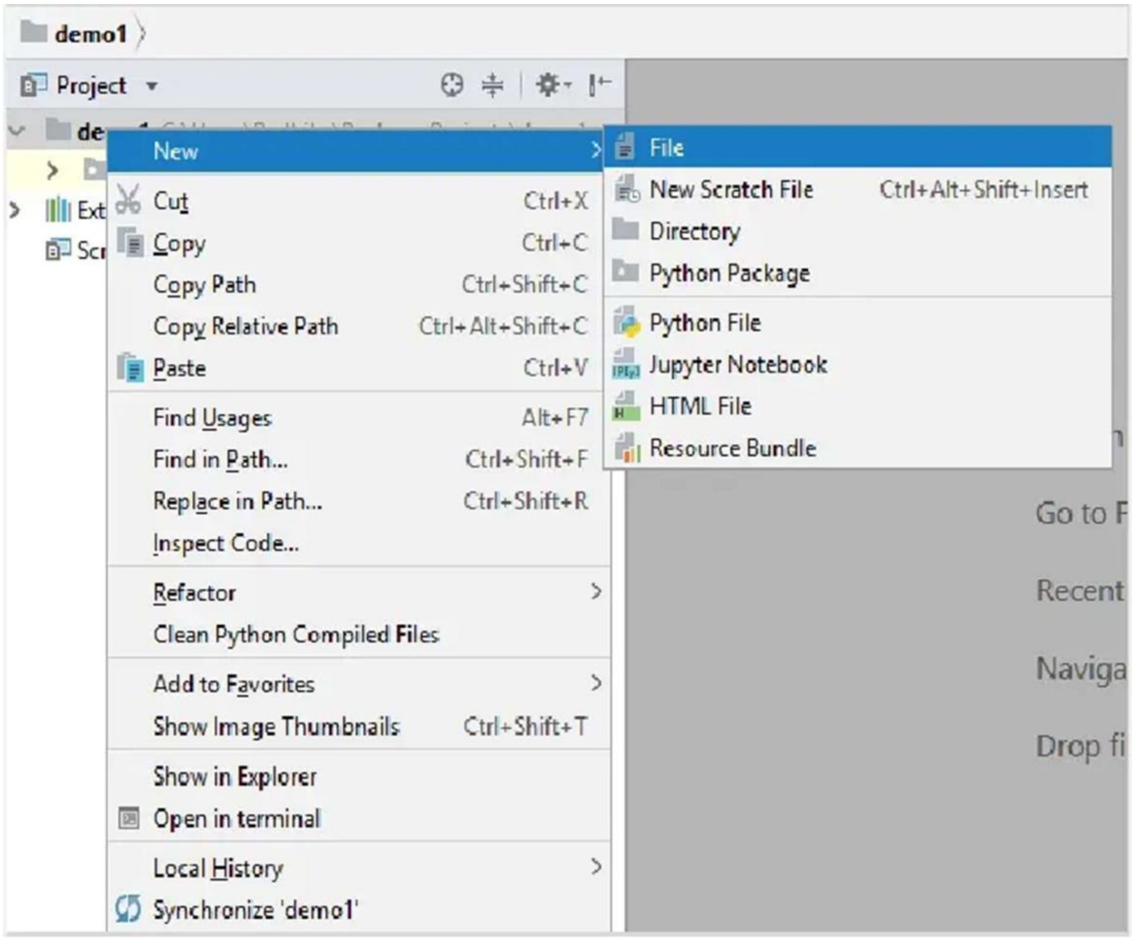
PYCHARM-UNDERSTANDING BASICS:

This chapter will discuss the basics of PyCharm and make you feel comfortable to beginworking in pycharm editor:

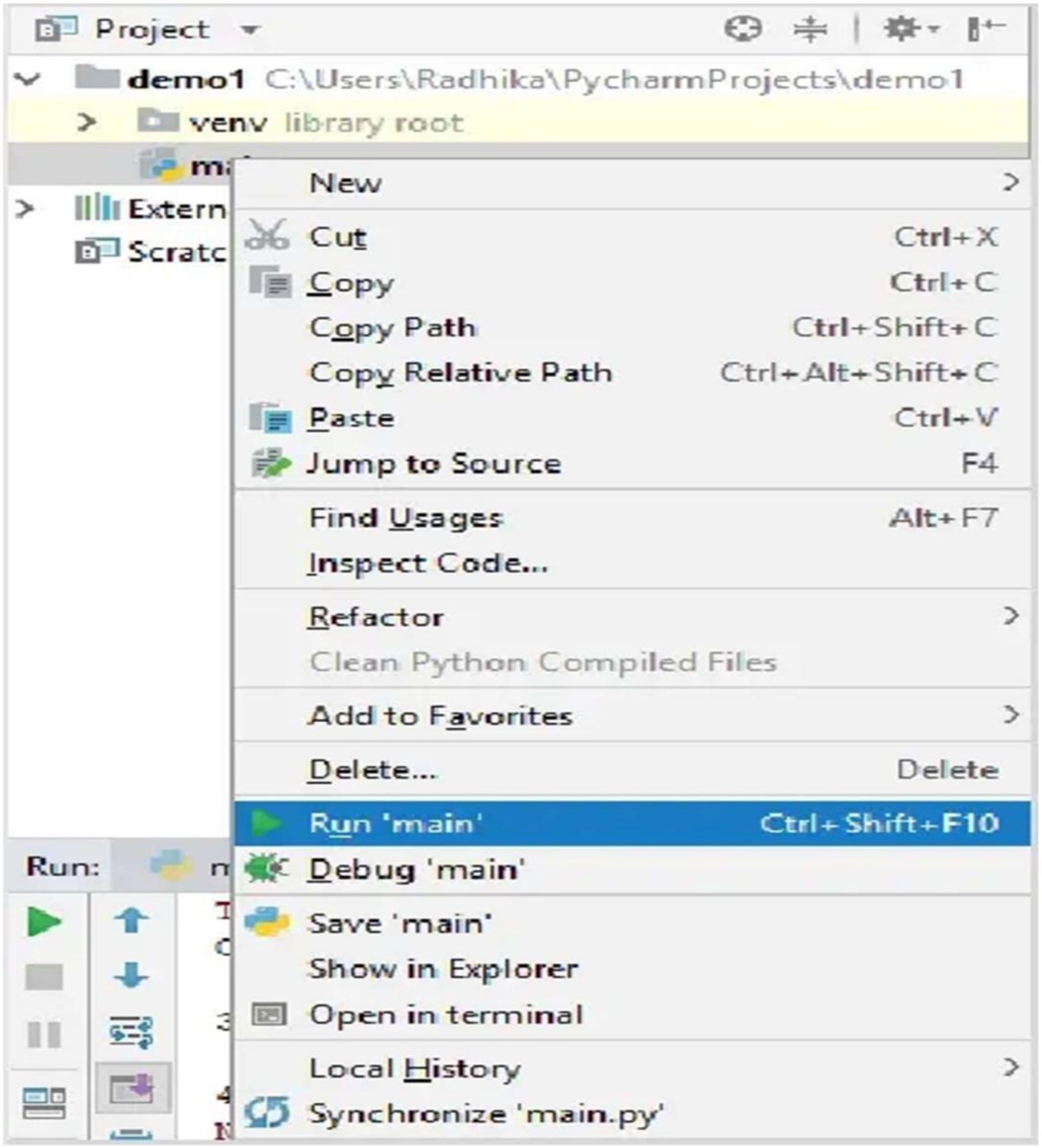
- When you launch PyCharm for the first time, you can see a welcome screen with entry points to IDE such as: Creating or opening the project
- Checking out the project from version control
- Viewing the documentation
- Configuring the IDE



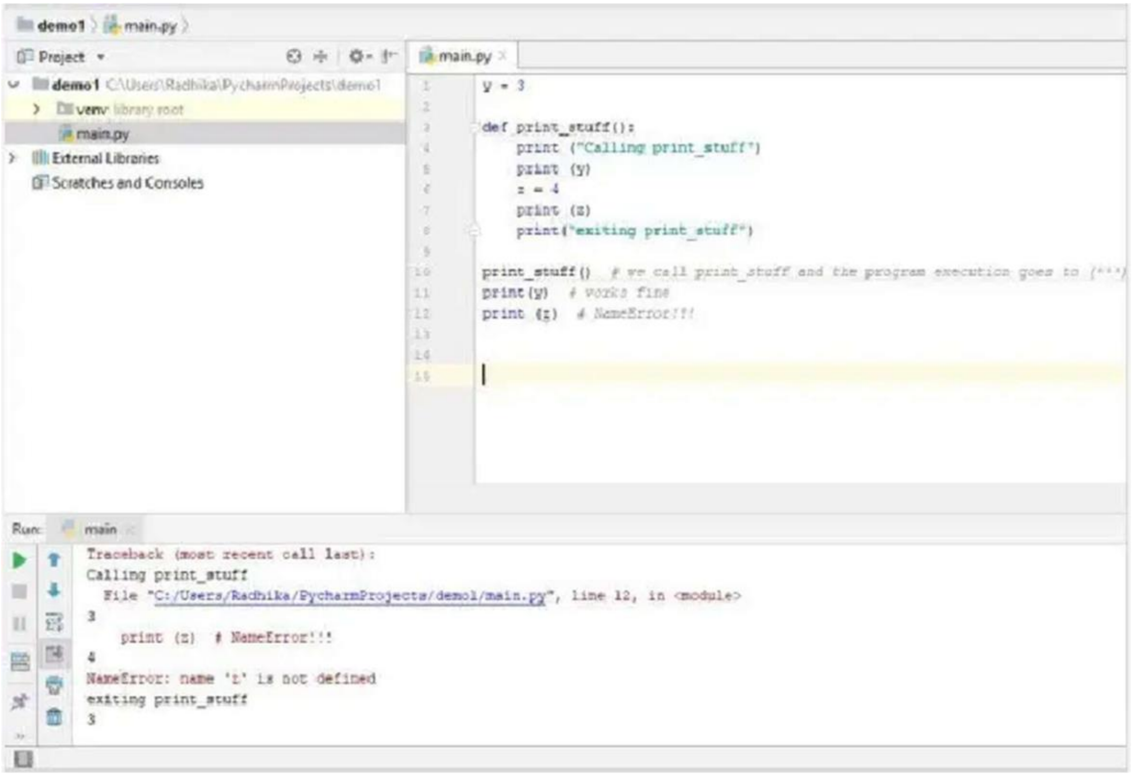
Recall that in the last chapter, we created a project named
demo1
and we will be referring to the same project throughout this tutorial. Now we will start creating new files in the same project to understandthe basics of PyCharm Editor



This code can be run within IDE environment. The basic demonstration of running a program isdiscussed below:



Note that we have included some errors within the specified code such that console can execute thecode and display output as the way it is intended to



```
y = 3

def print_stuff():
    print ("Calling print_stuff")
    print (y)
    z = 4
    print (z)
    print("exiting print_stuff")

print_stuff()  # we call print_stuff and the program execution goes to (***)
print(y)  # works fine
print (z)  # NameError!!!
```

```
Traceback (most recent call last):
Calling print_stuff
  File "C:/Users/Radhika/PycharmProjects/demo1/main.py", line 12, in <module>
3
    print (z)  # NameError!!!
4
NameError: name 'z' is not defined
exiting print_stuff
3
```

FEASIBILITY STUDY:-

game involves a player shooting at aliens with a laser cannon while they descend and move left and right. The goal of the game is to survive as long as possible and accumulate as many points as possible.

The feasibility study for a Space Invaders game would involve an analysis of the development costs, market potential, and technical Space Invaders is a classic video game that was released in 1978. The requirements.

Development Costs: A feasibility study would need to include an estimate of the costs associated with developing and releasing a Space Invaders game. Costs such as game design, programming, and artwork would need to be considered. Additionally, marketing costs, suchas advertising, would need to be taken into account.

Market Potential: In order to determine the potential success of a Space Invaders game, a feasibility study would need to consider the size of the current market for this type of game and the potential for growth. This would involve researching the popularity of Space Invaders and other similar games, as well as the potential for expanding the market.

Technical Requirements: A feasibility study would need to consider the technical requirements for the game, such as the hardware and software needed to run the game. This would include the hardware and software needed by the developer, as well as the hardware

Approach:-

- Import the required module.
- Initialize the pygame.
- Create three functions:
    - **isCollision():** Which tells us whether the collision has occurred or not?
    - **game_over():** Which returns True or False on the basis of which the codedecided if the game has ended.
    - **show_score(x, y):** This shows the score on the screen
- Create an infinite loop to execute the code continuously.

isCollision():-

It's very simple actually. Before explaining this, we want you to take a look at the collisionportion of the code inside the game loop first below.

The criteria for collision set inside the function is the simplest thing as the distance between the bullet and the invader (our enemy). As you can see the formula used for calculating distance is something that every student study in their high school mathematics class. It's the formula of the distance between two points having coordinates (x1, y1) and (x2, y2) which are being passed as parameters of the isCollision() function.

Here, we have set the criteria that if the value of the distance is less than or equal to 50, then it means collision has occurred. The value is chosen on the basis of the height and width of the png image used for the bullet and the invader. The value can be tweaked as per your ownrequirements by using the trial and error method.

So, whenever the position of the bullet and the invader changes then the isCollision() function checks if a collision has occurred or not. That is the reason why it is being calledinside the game loop.

game_over():-

Which returns True or False on the basis of which the code decided if the game has ended. For understanding the game_over() function, let's take a look at the below snippet of code which is present inside the game loop

Before getting into the explanation of code, it is recommended to know about the coordinatesystem followed in pygame. Take a look at the image below:

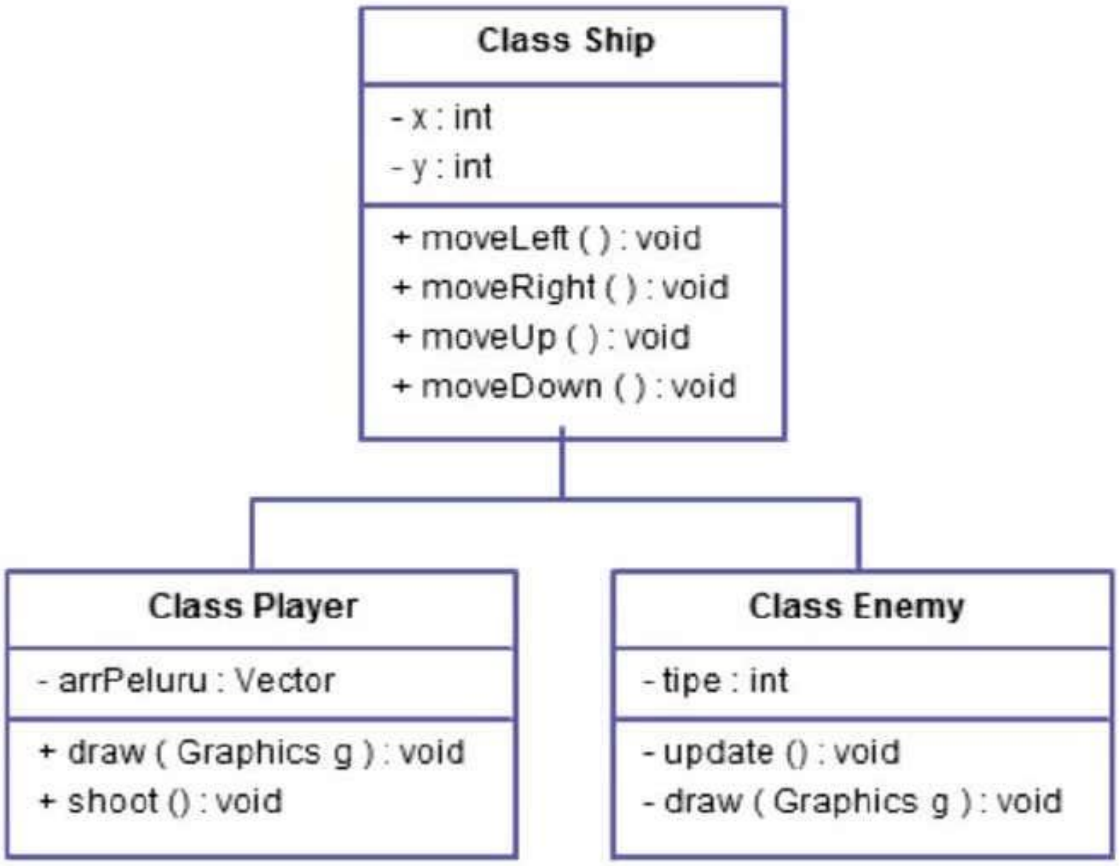
So, the criteria for game over is also collision. When the y-coordinate of the invader is greater than the spaceship i.e., 450 (y-coordinate of the spaceship), and the distance between the invader and the spaceship is less than 80 then a collision occurs and the game_over() function is called followed by the explosion sound.

show_score(x, y):

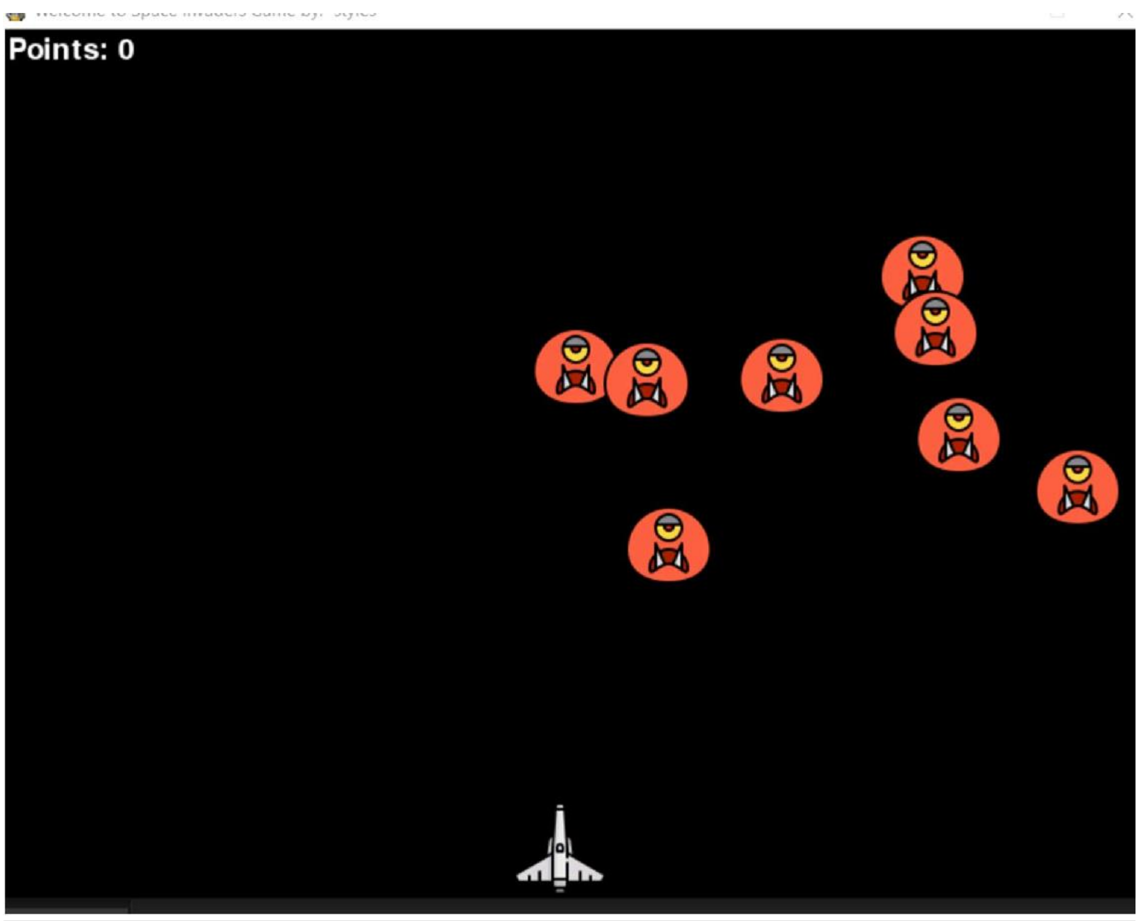
The only thing show_score() function is doing is showing the score on the screen in a properfont selected by the user.

Every time a collision between the bullet and the invaders is happening a variable "**score_val**" is being incremented. This variable is then being displayed on the screen by theshow_score() function as can be seen in the above code snippet.

CLASS DIAGRAM :-



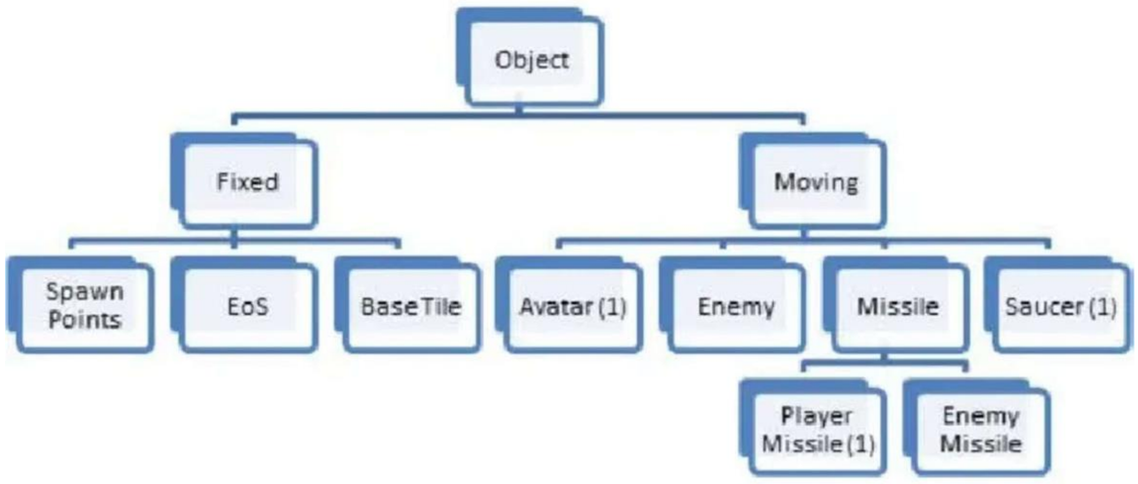
RESULT INTERFACE:-



CONCLUSION:-

This Online Space invaders game provides facility to paly space invaders game anywhere and anytime.It save time since user does not need to wait for result So all user get extra knowledge and skills.This project has helped us in getting a dearer understanding of real word application development. This is the part that I have mostenjoyed, because I really like programming and designing game logic. Additionally, coming up with differentEnemies and Boss logic provided me a fun challenge, and gave me the opportunity to learn many different issuesabout game design and development. Furthermore, I discovered many interesting things about developinga gamefor people with colorblindness disability, and found out a way for them tothoughtfully enjoy the game. On the otherhand, the obtained video-game can be playedat different operating systems and it has a very good performance evenon dated computers. Moreover, it can be easily ported to any of the current generation game consoles. Finally, Iwould like to highlight that most of the things done throughout this project were learned while developing it, as Ihad very little experience in the whole process of video-game development. I had made very simple video gameswithout using full fledged game engines, and never before had I designed any sound effect or graphic material.

OBJECT DIAGRAM:-



TEST ANALYSIS:-

Space Invaders is a classic arcade game that has been around since 1978. It is a shooter game where the player controls a ship and must shoot down rows of aliens before they reach the bottom of the screen. The game has a simple concept but requires quick reflexes and good aim to succeed.

The game has a number of features which make it attractive to players. Firstly, it is simple and easy to understand. The objective is clear and the controls are straightforward. Secondly, the level of difficulty increases as the game progresses, providing a challenge for players.
Finally, the game has an appealing retro look and sound which gives it an element ofnostalgia.

In terms of gameplay, Space Invaders is an engaging and enjoyable experience. Players mustmove their ship left and right while shooting down aliens with their laser cannons. The game also features power-ups which can be collected to gain an advantage over the aliens. The game has a fast-paced and intense atmosphere which keeps players on their toes.

Overall, Space Invaders is a classic arcade game which has stood the test of time. It is a simple yet challenging game which requires quick reflexes and good aim. The game has a classic feel and sound which makes it appealing to both old and new generations.

Limitations :-

1. Lack of Long-term Goals: One of the biggest limitations of the Space Invaders game is that it does not offer any long-term goals. Once the player has completed all the levels, the game is over. There is no incentive to keep playing or to come back to the game.

2. Limited Interaction: Another limitation of Space Invaders is that there is limited interaction between the player and the game. The player can move the ship left and right, and shoot at the enemies, but that’s about it. There is no way to interact with the environment or the enemies in any meaningful way.

3. Lack of Variety: The levels in Space Invaders are all quite similar, which can make the game feel repetitive. As the game progresses, the enemies get faster and the bullets move faster, but the overall structure of the game remains the same. This can lead to boredom after a while.

FUTURE SCOPE:-

In the future, space invader games could be further enhanced with more levels, more enemies, and more challenging levels. There could be online leaderboards and tournaments, allowing players to compete against each other. Additionally, the game could offer special power-ups or rewards for completing certain levels or achieving certain milestones. There could also be new weapons, such as lasers or missiles, to add a new layer of strategy. There could be more immersive graphics and sound effects to draw players in and make the game more exciting. Finally, there could be new game modes such as a cooperative mode where two players work together to defeat the aliens or a survival mode where the player must survive against an endless wave of enemies.

REFERENCES AND BIBLIOGRAPHY:-

- **BOOKS USED:**
  - Python programming version 3.7.4(Dr.A.Kannan,Dr.L.Sai Ramesh)
  - Software Engineering (Shalini Puri)
  - Software Engineering (Pressman)

- **SITES USED :**

- www.scribd.com
- www.wikipedia.org
- https://www.fatcon.com/search?word=space+invaders
- https://www.treepik.com/search?dates=any&torma=search&page=1&query=space%2 0background&sort=popular
- https://www.rapidtables.com/convert/color/hex-to-rgb.html
- https://github.com/
- https://www.mathplanet.com/educaton/algebra-2/conic-secons/distance-between-two- points-
- Pygame. URL: htp://www.pygame.org/hit.html.
- The ESA 2015 Report. URL: htp://www.theesa.com/wp-content/uploads/2015/04/ESAEssentalFacts2015.pdf .
- Video Games Industry sales. URL: htp://vgsales.wikia.com/wiki/Video_game_industry.

APPENDIX:-

- Code of solution :-

```
import pygame
import random
import math

pygame.init()
screen_width=800
screen_height=600


gameWindow=pygame.display.set_mode((screen_width,screen_height))
pygame.display.set_caption("Space Invader Game")
icon=pygame.image.load('spaceship.png')
pygame.display.set_icon(icon)

# backgound
backg=pygame.image.load('background3.png')
#player
playerImg=pygame.image.load('spaceship (1).png')
playerX=370
playerY=480

def player(x,y):
    gameWindow.blit(playerImg,(x,y))

#enemy
''' enemyImg = pygame.image.load('enemy.png')
enemyX = 370
enemyY = 50

enemyX=random.randint(0,736)
enemyY=random.randint(50,150)
enemyX_change=4
enemyY_change=40
'''
enemyImg=[]
enemyX=[]
enemyY=[]
enemyX_change=[]
enemyY_change=[]
no_of_enemy=6

for i in range(no_of_enemy):
    enemyImg.append(pygame.image.load('enemy.png'))
    enemyX.append(random.randint(0,736))
    enemyY.append(random.randint(50,150))
    enemyX_change.append(4)
    enemyY_change.append(40)

def enemy(x, y,i):
    gameWindow.blit(enemyImg[i], (x, y))

#bullet cretion
bulletImg=pygame.image.load('bullet (1).png')
bulletX=0
bulletY=480
bulletX_change=0
bulletY_change=10
bullet_state="ready"

def fire_bullet(x,y):
    global bullet_state
    bullet_state = "fire"
    gameWindow.blit(bulletImg,(x+16,y+12))

def testCollision(enemyX,enemyY,bulletX,bulletY):
    distance=math.sqrt(math.pow(enemyX-bulletX,2)+math.pow(enemyY-bulletY,2))
    if distance<27:
        return True
    else:
        return False
score=0

fonnt=pygame.font.Font('freesansbold.ttf',32)
textX=10
textY=10

def show_score(x,y):
    score1=fonnt.render("Score :"+str(score),True,(255,255,255))
    gameWindow.blit(score1,(x,y))
text=pygame.font.Font('freesansbold.ttf',72)
def game_over():
    text=fonnt.render("Game over"+str(score),True,(255,255,255))
    gameWindow.blit(text,(200,250))
block_update=5
playerX_change=0
closewindow=False
white=(255,255,255)

while not closewindow:
    gameWindow.fill(white)
    gameWindow.blit(backg,(0,0))
    for event in pygame.event.get():
        if event.type==pygame.QUIT:
            closewindow=True

        if event.type==pygame.KEYDOWN:
            if event.key==pygame.K_LEFT:
                playerX_change=-block_update
            if event.key==pygame.K_RIGHT:
                playerX_change=block_update
            if event.key==pygame.K_SPACE:
                if bullet_state=="ready":
                    bulletX=playerX
                    fire_bullet(bulletX,bulletY)

        if event.type == pygame.KEYUP:
            if event.key == pygame.K_LEFT or  event.key==pygame.K_RIGHT:
                playerX_change = 0
    playerX+= playerX_change

    if playerX<=0:
        playerX=0
    elif playerX>=736:
        playerX=736

    #enemy movement
    for i in range(no_of_enemy):

        #when the game is over
        if enemyY[i]>440:
            for j in range(no_of_enemy):
                enemyY[j]=20000
            game_over()
            break
        enemyX[i]+= enemyX_change[i]
        if enemyX[i]<=0:
            enemyX_change[i]=4
            enemyY[i]+=enemyY_change[i]
        elif enemyX[i]>=736:
            enemyX_change[i]= -4
            enemyY[i]+=enemyY_change[i]
        collison = testCollision(enemyX[i], enemyY[i], bulletX, bulletY)
        if collison:
            bulletY = 480
            bullet_state = "ready"
            score = score + 1
            enemyX[i]=random.randint(0,736)
            enemyY[i]=random.randint(50,150)

        enemy(enemyX[i], enemyY[i], i)

    #bullet movement
    if bulletY<=0:
        bulletY=480
        bullet_state="ready"

    if bullet_state=="fire":
        fire_bullet(bulletX,bulletY)
        bulletY-=bulletY_change

        #print(score)
    player(playerX,playerY)
    show_score(textX,textY)
    pygame.display.update
```