Pac-Man in Space

Henan Mu ([mu.he@northeastern.edu](mailto:mu.he@northeastern.edu)), Prachi Deepak Kunjir (kunjir.p@northeastern.edu),

Sai Saran Pathuri (pathuri.sa@northeastern.edu),

Sai Siddhartha Kondamu ([kondamu.s@northeastern.edu](mailto:kondamu.s@northeastern.edu)), Yuxuan Cheng (cheng.yuxu@northeastern.edu)

Information Systems, Northeastern University

Software Engineering Systems, Northeastern University

***Abstract*: This project intends to provide an end user computer game using JavaFx for anyone to enjoy. We created a known game among children and even among adults which is known as Pac-Man! This is geared towards developing an end user game with the objective of utilizing and Java object-oriented programming concepts while exploring JavaFx and keeping implementation fun and interesting. The program has many features and rules. Dashboard presents six buttons as you land on home page are listed as: Player, Play, Scores, Help, Credits, Exit. Player’s option provides you to give names to your character as per your choice. Play will take you to main game pain where the game starts. Game is pretty simple to understand and play. You have given 3 lives/chances. You can move Pac-Man to right side or left side. The game as soon as it launches, will randomly generate stars, ghosts and meteoroids moving vertically. You need to move your Pac-Man either to right and left as you need to collect stars and avoid hitting meteoroids and ghosts. You will receive points as you collect more stars. If you hit meteoroids or ghosts, your one life will be taken amongst three! If you hit either the ghost or the meteoroids for the 3rd time then you will be taken to the dashboard and your score will be saved in a file. You can see your scores in scores section from the dashboard. If you are new in this game then you can head over to the Help section and read the instructions on how to play this game. Credit section gives out names of the developers of the system. Exit section will take you out of the game and will close both the main and sub panes. This program implementation majorly focused on Class Definition, Inheritance and Polymorphism, Interfaces, Collections and Iterators, Lists, Sets. The game can be used on both the mac and windows system.**

***Keywords- (***Dashboard , Game stage, Pac-Man, Star, Ghost, Collision, OOPS concepts, Inheritance and Polymorphism, Interfaces, Collections and Iterators, Lists and Sets, FileI/O, Animations, Stage, Scene, Layout, Button, Label, SubScene, VBox, Images, ImageView )

# **I. Problem Description**

To incorporate all of the OOPss concepts with JavaFx that we learned throughout the course, we wanted to create something entertaining and engaging. We thought there was no better approach than to create a game that everyone enjoys. ​In the process of game creation, we want to explore more in terms of how to create and apply animations that would be visually appealing to the user that is engaged in the game. While we believe that it is important to develop applications to build for business it is even more important to build something that is memorable for longer persistence.

# **II. Analysis (Related Work)**

The classic and wildly popular action maze chase video game Pac-Man was first released in Japan. In this game, the player guides the yellow Pac-Man character through an enclosed maze while avoiding ghosts to consume all the dots put there.

Pac-Man is one of the most popular arcade games of all time. In America, it brought in more than $1 billion over three months. selling more than 100,000 arcade units in just two years. As soon as it was released, it was a big success all throughout the country.

However, there are some shortcomings about Pac-Man. The game is very straightforward and basic. It’s an old game. Continually playing the game gets boring. Although it has evolved over the years, it is still the same game that came out in the 1980s.

A picture containing background pattern

Description automatically generated

Figure 1. In-game screenshot of the classic Pac-Man.

# **III. System Design**

We incorporate all the OOP ideas with JavaFx that we learned throughout the course to design the application. Users play with our JavaFX application and the application interacts with some text files which are used to do data persistence.

Text

Description automatically generated

Figure 2. System architecture

In the Pac-Man in space application, we have two pages which are menu pages and game pages:

A picture containing graphical user interface

Description automatically generated

Figure 3. Draft UI designs of menu and game pages

When a user starts the game, user goes to the menu page firstly. In this page there are some buttons including CREATE, PLAY, SCORES, HELP, CREDITS and EXIT. PLAY button is used to start the game which means going to the game page. CREATE is used to submit a username and EXIT button is used to close the application. When user click EXIT button, we save the user’s scores to a specific file and then close the application. Other buttons are used to present some specific information like instructions, history scores and so on.

Diagram

Description automatically generated

Figure 4. Workflow of Pac-Man in Space

Here is the Component diagram of the application.

Diagram, schematic

Description automatically generated

Figure 5. Component diagram

# **IV. Implementation**

To have a brief overview, we have defined the following classes:

In the dashboard stage, we have, custombutton which inherits class button, customsubscene which inherits class subscene, customsmall label which inherits class label, custombig label which inherits class label; In the game stage we have, Pac-Man, userscore, userscore label which inherits class label.

The front stage part and the game stage part will be discussed separately as followed.

1. main method

In this method, we override the start method only. That is to say, we only create a new game dashboard (That is, the main stage) object and assign it to the parameter ‘primarystage’ by the dashboard.getMainStage method which returns a stage type value. Then we call primarystage.show.

2.Dashboard (main stage)

2.1 Initialization: The main stage and the main scene.

Initially, this class defines and stores a Stage - mainstage, a Pane - mainpane, and a Scene - mainScene. The mainpane includes all the contents including - all the widgets including buttons and subscenes (which will be discussed later), and it is in the main scene. The mainscene is in the mainstage using the mainstage.setscene. The mainstage has a function: getStage that returns this mainstage to the main, as mentioned in part 1.

2.2 Create name button: This part is implemented by creating two classes: the custom button and the custom subscene. The customersubscene inherits the subscene class [7]. Based on the subscene’s constructor (which creates an empty subscene), it defines the outlook of the subscene - it adds a background picture, and set it to a default hidden position. When the respective button is clicked, it will show up / hide by moving a specified position with a transfer effect. This is implemented by creating a movesubscene method which judges whether to show or hide the current page. Then create a translatetrasition object, and use the setduration to set the transition display time to 0.3 seconds.

The customerbutton inherits the button class [6]. Similar to the customersubscene class - based on the button’s constructor (which creates a new empty button), it defines the outlook of the button - it adds a background picture, add font type and size. In addition, it adds an eventhandler and overrides the handle method. When clicked on/released, the button will have an x-axis move down/up, as if the button is really ‘pressed and released. (Meanwhile, the moveSubScene should be called on the particular subscene, but it is never defined in the class.)

In this part, a button and a corresponding subscene are created, so that clicks on the button will call that moveSubScene method in the subscene. The subscene part includes a text field and a submit button (this button is in). The logic is, a string variable playername is initialized to be ‘Player’ when the game dashboard object creates. If no submission is made then the player’s username will be saved as ‘Player’; otherwise, the name would be what the user types in. This is implemented by adding another button event handler to submit button to set the playername to what the user types.

2.3 Show credits/ Player Scores / Help SubScene:This part creates buttons and respective subscenes. For the highmarks part, the differences are, it reads a file at a relative address and displays the files by lines. This is implemented by creating a method named ‘Readfromfile’. This method throws a file-not-found exception when the file not exists (which is the only possible exception as we recognized). Then create a new file object, a scanner object, and a new ArrayList named ‘userslist’. apply the scanner to read line by line and use userslist to store them. At last, the method returns the whole userslist. After receiving the userslist, we create a 10-size Vbox to store them. Then we create an iterator of the userslist. Every time the iterator moves, create a userscorelabel to store it. The user score label class inherits the label class [8]. Based on the label’s constructor which creates an empty label, it defines the background picture of the label and font of the text. The credits & help part has a similar framework, and they display staff name & game tips respectively.

2.4 Exit button: The exit button is created individually without subscenes. The button’s event handler is overridden to shut down the whole program & submit a record of the current player (no matter name is modified or not!) when clicked.

To specify how the record is submitted and written into the file, we define a method called ‘savetofile’. In this method, we define a file object with a relative address and a file writer object. A boolean variable in the game dashboard is initialized to be false; after appending the record ‘name - score’ to the end of the text file, the boolean variable will be set to true so that no repeat actions will be made.

2.5 Play button: Similar to the exit button, this button has no corresponding subscenes. The event handler was overridden as followed: when the button is clicked, create a new Pac-Man screen object and then call its createNewGame function, which calls most of the main functions defined in Pac-Man screen class, which will be discussed later.

3. Game stage

3.1 constructor

When a new instance of this class is created, the keyboard listener and framework are created. The constructor contains two main functions: initializeStage and createKeyListeners. The initializeStage method will create the framework widget including an anchorpane named ‘gamepane’, a scene named ‘gamescene’, and a stage [9] named ‘gamestage’. Just like the dashboard stage, the anchorpane is in the scene, and the scene is in the stage.

The createKeyListeners method creates a listener of the keyboard as we need users to control the character's movement via arrow keys. As a result, we create separate event handlers for keycode.right and keycode.left - these key pressed would set isleftkeypressed (initialized boolean value) state to true; and vice versa - when the key is released the value will be set to false. These settings are convenient for later operations because we need to set both angle and movement of the Pac-Man.

3.2 Createnewgame : This method will set everything ready for the user to run the game, including the rolling background, all items (objects) that should appear, the general and constant logic in this game (such as how Pac-Man life reduces, how to tell whether a collision happens, how to reuse ghost objects after it is not seen), and the background of the game. In addition, it hides the dashboard stage and shows the gameplay stage using this.menuStage.hide and gameStage.show;

Methods called here will be discussed as followed:

3.2.1 createGameBackground method

When this method is called, 2 separate gridpanes are created. Then Images are filled in these grid panels by creating new image views. After these, the second gridpane is set to -1024 on the y-axis.

3.2.2 createPacMan method

When this method is called, a Pac-Man object will be generated. The Pac-Man will be given a picture and a position. Then Pac-Man will be added to the gamepane.

3.2.3 createGameElements method

This method will create all the required items that appeared in the game. The integer playerlife variable will be assigned to 2 (that is, can still alive after 2 hits, which means the player has 3 lives in real).

Create a star object for the player to catch, with the proper image and proper position by applying setNewElementPosition. This method sets the element’s y layout to a random high position (such as -1024). Then add it to the gamepane.

Create CustomSmallLabel named ‘pointlabel’ (just like userscorelabel that inherits the label, this class sets another form of background and font). Then add it to the gamepane. Use an array of imageview with length 3 named ‘playerlifes’ to store life stars. Use a loop to print as many stars as the amount of variable ‘playerlife’. Then add them to the gamepane. Similarly, create an array of imageview with length 3. Then use a loop to initialize (create) 3 ghosts. Set these ghosts with a position and an image. Add them to the game pane. Repeat this whole paragraph to create another 3 ghosts in a different color.

3.2.4 createGameLoop methodCreate a new animationtimer object to run the timeline (it is just like a while loop but could easily end without a break statement). Add the following methods to the timeline:

3.2.4.1 moveBackground: Let the 2 gridpanes defined in the background (in part 3.2.1) move downwards as if the Pac-Man is moving in the space. This is implemented by setting the y-layout to add 0.5 each time the method is called.

3.2.4.2 moveGameElements: Set the star and ghosts to fall as if the Pac-Man is turning forward in outer space. This is implemented by setting the layout of the star & ghosts to add 5 each time the method is called.

3.2.4.3 checkIfElementAreBehindThePacManAndRelocated: Check if the star & ghosts exceed the downside edge of the screen. If they do, set their y-layout to a high position by applying setNewElementPosition, as mentioned in 3.2.3.

3.2.4.4 checkIfElementsCollide : This method checks if the Pac-Man(user) collide with ghosts/star. If hits star, scores will be added by 5; if hits ghost, playerlife will be minus by 5. To implement, the key judgment is to calculate the distance between the center of the Pac-Man and the object(let’s say, it is the ghost or star here). If this distance < object’s radius + Pac-Man’s radius then events happen.

3.2.4.5 movePacMan: This method allows Pac-Man to move & change its forward angle. This is implemented by detecting the keyboard.left & keyboard.right state.

To specify, if both left and right are pressed/ released, then the angle will not change and it will not move to left & right; if only one key of them is pressed then the angle will turn to that direction by 5 degrees, and it will move to that direction by 5.

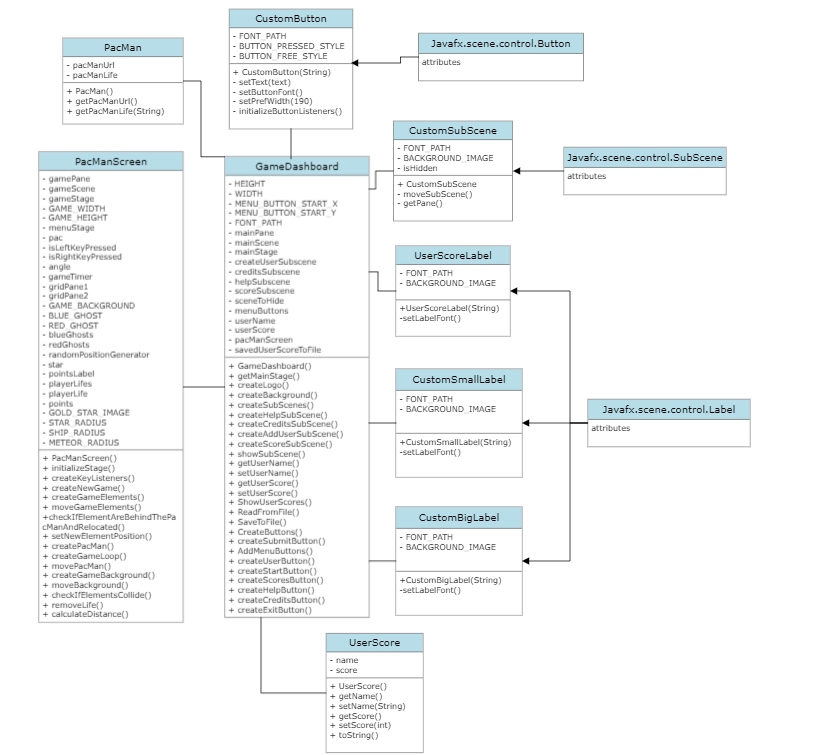


Figure 6. UML Diagram

# **V. Evaluation**

The explanation and sample run screenshots:

When the player starts the game, the game home screen welcomes the user with its intuitive and modern UI as shown below

Graphical user interface

Description automatically generated

Figure 7. Game Home Screen

Every element of this UI is meticulous crafted using Javafx Scenes and subscenes for the smoother transitions

To guide the player, HELP tab is created. This enables the players to enjoy the game without finicking with the controls. This below picture can demonstrate it function.

Graphical user interface, text

Description automatically generated

Figure 8. Help tab

Before starting the game, player can provide “player name”. This player is recorded in a text file, which is further used for the Game Scores.

Graphical user interface, website

Description automatically generated

Figure 9. Player Tab

Playing the main game is always fun, so, let get into the main part! When the player clicks on the “PLAY” tab it will instantly take you to the game screen. You can see how the real fun looks like in the below picture.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 10. Space Pac-Man Game screen

Whenever Pac-Man catches a “Star” the points increase while avoiding the ghosts, when you hit a ghost, your life decreases and its game over when you lose your 3 lives.

Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated

Figure 11. Collision between Pac-Man, star and ghost; increment points and decrement lives

When the game ends the user can always check his scores in the **Scores** tab as tested below.

Graphical user interface

Description automatically generated

Figure 12. Scoreboard Tab

The credit section shows our team credits as shown

Graphical user interface

Description automatically generated

Figure 13. Credits Tab

**Comparison between our solution and other people’s work:** The Pac-Man is an ageless classic, carrying the legacy of its own. There is no denying that classic is fun to play even in today’s Gen-Z era. We are not comparing our work to the classic; we want to take the concept of Pac-Man to the new and fun horizon.

Graphical user interface

Description automatically generated

`

Figure 14. Comparison between the native Pac-Man to our proposed solution

**User Testing and Reviews**: This game undergone multiple revisions and tests by real users. Three people were randomly picked at the Snell Library, Northeastern University and were given questionnaire, based on test result few drastic changes were done to the game-mechanics. After few repetitions, another batch of people were picked for the testing, and these are the real reviews we have collected.

*This is some real fun I had after long time, this game looks legit and invokes the memories that I’ve experienced in my childhood. I used play Pac-man in cartridge TV videogame. Thanks’ for asking to review this.*

*-Ankur, Abhinav (MSIS).*

*Pac-man in space is a crazy idea to pull off. I love this game, it’s awesome. I never thought we can make games like these for final projects. I’m a Bio engineering grad, if I’ve known that I can make game’s and learn, I’ve could a picked CS or IS as my major. Too bad, it’s already late.*

*-Prem Varma (MS Bio Engineering)*

# **VII. Discussion (Reflection)**

The results and the data or outcome of our project; are provided with some insightful discussions in this section.

**The Idea:** The objective of the project “Space Pac-Man” is to learn and demonstrate the concepts of Object-Oriented design in an exuberant way. It is started way before the idea of Pac-man or before the idea of a game. Collaboration is always the key for as after multiple brainstorming session and confirming that all the possible concepts and beyond are covered using JavaFX with fun. This gave birth to “The Space Pac-Man”.

**The Challenges:** Building the in Game-Mechanics is quite a challenge. As it not only requires the mastery on the OOD concepts, it also demanded more and beyond from the all the team members. After spending countless. After setting some elbow grease, most of the game mechanics was tweaked and improved including pace of the ghosts, Pac-man, movements etc. in the process we learned the process of controlling the **Frames-Per-Second (FPS)** in the game.

One of the main hurdles this project encountered is the intuitiveness of the User Interface. As it is the facade of the application, it should be as modern as possible without sacrificing the fluidity of the game or animations. It took countless hours of research, which ended up in learning of uncharted approaches of JavaFX called subscenes.

**The Outcome:**

Apart from learning and exploring the concepts like

* Class Definition​ [1]
* Inheritance and Polymorphism​ [2]
* Interfaces[3]
* Collections and Iterators​ [4]
* Lists​ [5]
* Sets​ [5]

This project made us explore the uncharted concepts of JavaFX like **javafx.animation.\*** [10] class along with mastering the very concept of **Animationtimer** and its manipulation techniques.

Mathematical equations were used to determine the predict the and calculate the co-ordinates of the in-game elements.

The project cannot see it’s full potential unless the team members collaborate synchronously by orchestrating the plans perfectly together. **Collboartion** is a key skill have handy not only for this project, in real-time job scenarios too

# **VIII. Conclusions and Future Work**

In this project, we have come up with a game that is everyone’s favorite i.e., Pac-Man using JavaFx and all the concepts that we learned from the Object-Oriented Design Course. We mainly concentrated on changing the usual Pac-Man to give it a new look and feel so as to make it more interesting and at the same time storing the player’s score in a local file once a game is completed.

The following are the problems we found but not yet explored in the project. They are:

* We have built this game such that only one player can play the game at a time.
* The game can only be played at only one complexity level.
* We are using files to store user scores as a result of which the scores are not reflected immediately so we have to run the game once again to view a player’s score.
* Once a player loses all the three lives that he gets initially, we aren’t allowing the player to continue to play.

We as a team feel that we can still improve the game and take it to a level higher. The following are the improvements that we would like to do to this game in future:

* We would like to make it a multi-player game in which more than one player can play at a time.
* We also would like to let the user pick their favorite Pac-Man color before they start playing the game.
* As mentioned earlier, we would like to integrate the game with a database to store the information generated from the game which includes storing scores, results from tournaments and rewards won by players. This would make the user score details readily available in the game.
* We also would like to improve in terms of adding animations and sound effects to the game a little more.

# **IX. Job Assignment**

* Henan Mu: Component diagram, worked on game dynamics, mainly concentrated on collision between Pac-Man and stars, also worked on Credits subscene in the dashboard
* Prachi Deepak Kunjir: Created moving background for the game and worked on moving the Pac-Man according to keys pressed, and also the Help Section in the dashboard.
* Sai Saran Pathuri: Worked on game dynamics, mainly concentrated on collision between ghosts and Pac-Man; researched about assets online to match our requirements
* Sai Siddhartha Kondamu: UML Diagram, customized labels, button, subscenes, and ImageViews for the game and worked on generating ghosts at random intervals at random positions
* Yuxuan Cheng: Worked on reading player names and updated scores in the file, and dynamically updating the scores and lives of player and PacMan respectively in the game.

##### **X. References**

1. https://www.geeksforgeeks.org/classes-objects-java/.
2. https://www.codecademy.com/learn/learn-java/modules/learn-java-inheritance-and-polymorphism/cheatsheet.
3. https://www.geeksforgeeks.org/interfaces-in-java/.
4. https://www.geeksforgeeks.org/iterator-vs-collection-in-java/
5. https://www.geeksforgeeks.org/difference-between-list-and-set-in-java/
6. https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/Button.html
7. https://docs.oracle.com/javase/8/javafx/api/javafx/scene/SubScene.html
8. https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/Label.html
9. https://docs.oracle.com/javase/8/javafx/api/javafx/stage/Stage.html
10. https://docs.oracle.com/javase/8/javafx/api/javafx/animation/Animation.html