BANG BANG PEW PEW FINAL PROJECT REPORT

Team Nerds

Stanley Nguyen (stanguye), Khang Le (khang), Kawsar Ahmed (ahmedk), Zaiyan Muhammad (monem), Mayank Yadav (mayanky)

Department of Engineering, Boston University

EC 327: Intro to Software Engineering

Dr. Gianluca Stringhini

May 4, 2022

Github Link

https://github.com/Khangxlei/BANG-BANG-PEW-PEW

Youtube Link

https://youtu.be/CAwbjxbcbks

Software Architecture

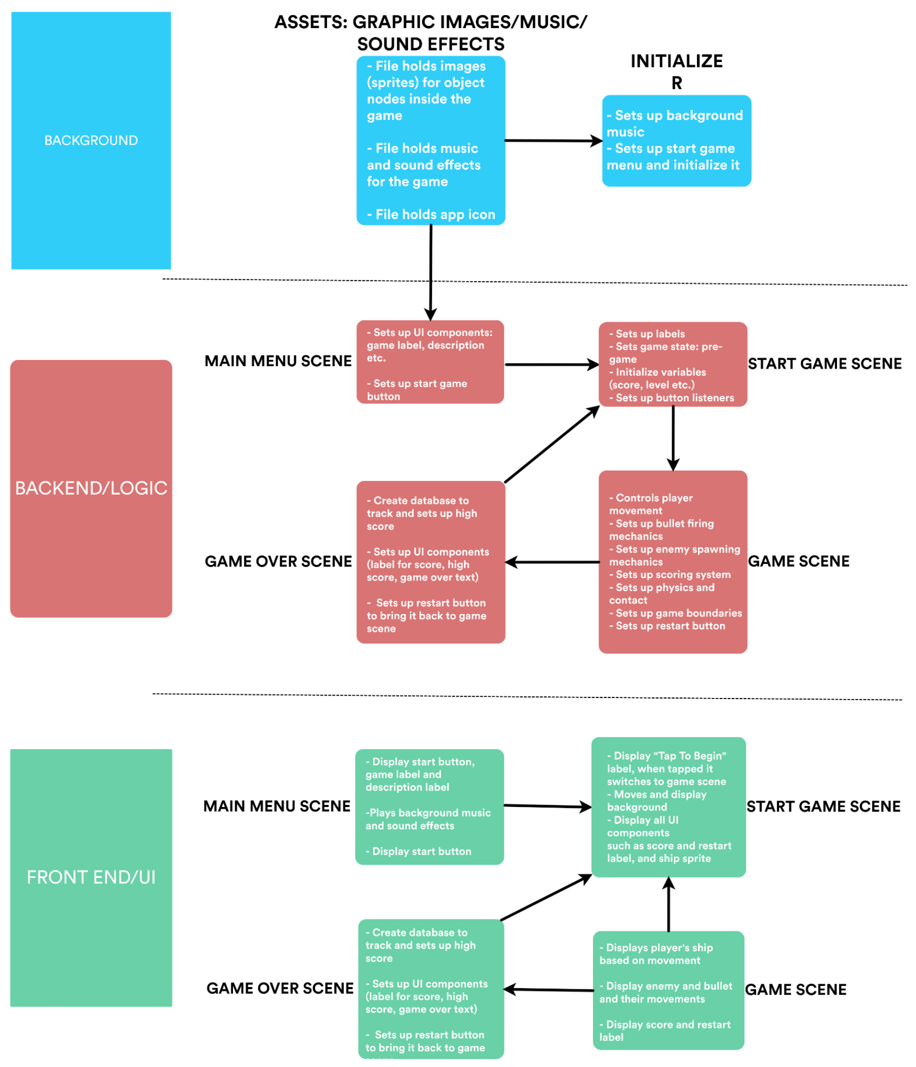
BACKGROUND
ASSETS: GRAPHIC IMAGES/MUSIC/ SOUND EFFECTS
- File holds images (sprites) for object nodes inside the game
- File holds music and sound effects for the game
- File holds app icon
INITIALIZE R
- Sets up background music
- Sets up start game menu and initialize it
BACKEND/LOGIC
MAIN MENU SCENE
- Sets up UI components: game label, description etc.
- Sets up start game button
START GAME SCENE
- Sets up labels
- Sets game state: pre-game
- Initialize variables (score, level etc.)
- Sets up button listeners
GAME OVER SCENE
- Create database to track and sets up high score
- Sets up UI components (label for score, high score, game over text)
- Sets up restart button to bring it back to game scene
GAME SCENE
- Controls player movement
- Sets up bullet firing mechanics
- Sets up enemy spawning mechanics
- Sets up scoring system
- Sets up physics and contact
- Sets up game boundaries
- Sets up restart button
FRONT END/UI
MAIN MENU SCENE
- Display start button, game label and description label
-Plays background music and sound effects
- Display start button
START GAME SCENE
- Display "Tap To Begin" label, when tapped it switches to game scene
- Moves and display background
- Display all UI components such as score and restart label, and ship sprite
GAME OVER SCENE
- Create database to track and sets up high score
- Sets up UI components (label for score, high score, game over text)
- Sets up restart button to bring it back to game
GAME SCENE
- Displays player's ship based on movement
- Display enemy and bullet and their movements
- Display score and restart label

Figure 1: Overview of our Software Architecture

The software architecture for our application consists of three sections: background, backend, and front end. For the background, it consists of the necessary, back-bone components for the game, which contains the assets files for the sprites, nodes, background music, and sound effects. Those are then used with the game view controller, which is used to initialize the game and display the screen that the user first sees in the game. The asset files will also be used in our backend section, where we have multiple files interconnected with each other, each being a scene on the iOS screen and behaving with each other. There is a main menu scene and a button that brings you to the game scene, and if you die it will display the game over the screen and a restart button that brings you back to the start game scene. The backend section is in charge of managing and setting up all the looks and functionality of the UI elements. Lastly, the front end section takes what the back end has set up for them, and simply displays them on the iOS screen.

Overview

Bang Bang Pew Pew is a space-themed multi-directional shooting arcade game designed by the team *Nerds* for the Final Project for EC327: Introduction to Software Engineering. The user is allowed to control a single spaceship in an asteroid field. The objective of the game is to shoot down the asteroids as they come towards the spaceship and at the same time avoid being hit by any missed asteroids. The spaceship is equipped with a gun that shoots straight upwards and the user gains a point each time an asteroid is shot down. The game ends when the spaceship collides with a single asteroid. The game gets harder at two stages, the user reaches 25 points and again at 40 points. At these stages, the duration between each spawned asteroid decreases by a factor of 0.4. In addition, the movement speed of the asteroids increases by a factor of 2. The game does not end until the spaceship collides with an asteroid. The first player to ever reach 100 points will be awarded a GameStop Gift Card worth $60.

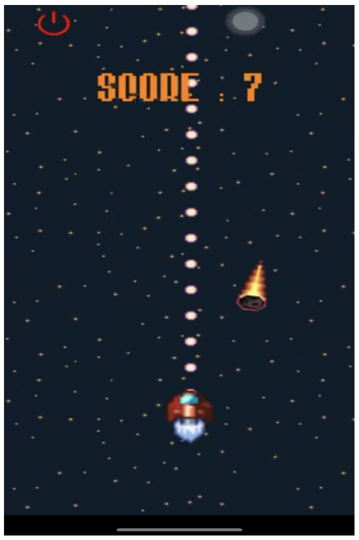

Figure 2: Game Screen

Marketability

The high-speed arcade shooter *Bang Bang Pew Pew* is ideal for users aged 10-40 years. This game is designed to improve cognitive abilities above all else as the user has to make split-second decisions in the harder stage of the game as it may not be enough to just avoid being hit by one asteroid. While dodging one asteroid, they must also make a decision of choosing the best new position. According to the software architecture, the asteroids are coded to target the spaceship at its last location. If the user dodges an asteroid but ends up in a bad position, it may easily hit the next asteroid approaching it.

For the older category of users, Bang Bang Pew Pew may aid in reducing stress levels and anxiety. As the game's difficulty does not periodically increase except for the two stages, the game is largely played at the same pace. Games of this sort are often developed by health professionals specifically to cater to stressed, anxious, and depressed patients. Moreover, the background music chosen for this game creates a dominantly positive ambiance that is meant to motivate and exhilarate the user.

Additionally, the game is also beneficial for improving reflexes and muscle memory for younger users.

Component Description p.1 (Front End)

While developing the game, we put a great emphasis on making the game easy to use. This was done by creating clear, straightforward, and distinct prompts for the user at every stage. After the game is run, the user is given the option to "Start Game". The prompt is color-coded in orange to distinguish between interactive buttons and visual text. Orange was chosen as it gives a clear contrast with the background. We chose a font called "Pixel" as it matched the theme of space and also felt age-appropriate for our target audience. The theme song was selected as it had a high index of happy and uplifting mood settings. Inside the game, the user has an option to restart the game by clicking the red icon on the top left. Again, this was chosen to be the color red as it is an interactive button but we wanted to have a more mellow contrast with the background so that it does not distract from the game. We chose a dynamic space background to make the game feel more realistic. The assets like the spaceship, asteroids, background, and music pack were taken from itch.io, a website that is popular for users to host, sell and download indie games.

Component Description p.2 (Back End)

As for the components within our code, we had 5 main files that are used for the final product, each representing a scene on the screen. The first file called “GameViewController”, which is not a scene, regulates everything that is running in the game, as well as initializes a screen that the user sees when they first open the app.

This then connects to our next file, called “StartScene”, which is the main menu of the game. This scene simply displays the name of the game, the objective, and a start button that brings us to the game screen.

The screen where all the action happens is called “GameScene”. The GameScene file sets up all the variables, such as scores and levels. It is in charge of showing a Tap to Begin label, which starts the game upon a user’s touch, all while displaying the sprite nodes such as the player’s ship, asteroids, bullets, and explosions. The file controls the gaming mechanics, such as setting up the firing bullet mechanic, enemy spawning rate and speed, and the spaceship’s controllability.

When the player dies, it will bring us to another file called “GameOverScene”, which is the file that keeps track of the high score, as well as displaying the score they got, their high score, and a restart button that will bring the user back to the GameScene file. The file keeps track of the high score by using a function that listens for a specific key, that is saved on the SwiftUI database, and by calling that key, the computer will be able to access the high score and display it.

Task Distribution

To make sure this project was done as efficiently as possible, we decided to assign each team member their respective roles with specific responsibilities they would be held accountable for. The project lead organized and facilitated the meeting between the members to discuss their various skillsets to evaluate who would best fit what role.

Khang: Technical Lead. He had extensive experience in programming and vast understanding of multiple programming languages. As technical lead, Khang wrote the code that is used to run and simulate the game. Setting up all the functionalities (i.e. bullets shoot when the user holds on the screen, explosions when there is contact with nodes).

Mayank: Documentation Lead. He proved himself to be very organized and was very comfortable with GitHub. Mayank organized the files that are needed to run the game, setting up the Youtube video.

Zaiyan: Interface Lead. We came to this conclusion because of his past experience in GUI programming. His main responsibilities were to ensure a visually pleasant, intuitive, and well-rounded experience for the user. Zaiyan went around to different parts of campus showcasing a demo of our app to get feedback on the game. Also, in charge of obtaining copyright-free sprites, images, and music.

Kawsar: Project Lead. Kawsar is a natural leader and can hold himself well. We entrusted him with the responsibilities of organizing and maintaining the schedule of all work that needed to be done for the project. Set up deadlines and worked with the other leads to make sure all the work was being completed on time. Made sure that the marketability requirements for the game were met.

Stanley: Specification Lead. Stanley was in charge of writing and running test cases, making sure that the game is behaving the way it is intended to. Also was responsible for commenting and organizing the code.

Assessment of % effort: Khang 24%, Kawsar 19%, Stanley 19%, Mayank 19%, Zaiyan 19%.

Timeline

- Discuss Ideas and make final decisions regarding the type of project
- Create objectives for the game
- Make the game
    - Create the main nodes (Bullets, spaceships, bullets, asteroids, and background)
    - Adding movement into the game
    - Create boundaries

- Adding in touch control
- Generating asteroids at random points periodically
    - Target user's current location
- Creating the shooting bullet feature
    - Bullets will be produced when holding and/or tapping the screen
- Adding game physics
- Implement Scoring
    - Increment the score by 1 after destroying an asteroid and track it throughout the game
    - Keeps track of high score
- Changing difficulty in-game
    - Score ranges were implemented to determine the spawning speed and falling speed of the asteroid
- Added an easter egg into the game
    - Can only be seen if a score of 100 is reached
- Implement a restart feature
- Create the main menu
- Implement music in the background (Non-Copyrighted Music)

- Run test cases
- Bug fixes
- Make improvements from feedback received from people playing the game
- Run final test cases and fix bug
- Sideload game into iPhone
- Create, Edit, and finalize the video of the game