<u>Progress Report</u>

- Increment 2 -

Group #7

1) **Team Members**

   -Mason Russo | mer21i | M-russ09

   -Tori Lee | tel21b | torilee7935

   -Parker Thompson | pjt21| PJThompson715

   -Jacob Sonn | jrs20k | jsonn32

2) **Project Title and Description**

   Pixel Pummel is an 8-bit arcade-style street fighter combat game. Players can step into the shoes of our unique fighters and face off head-to-head in varying environments. This game supplies an action-packed and nostalgic experience for all to enjoy.

3) **Accomplishments and overall project status during this increment**

   In this increment we tackled the challenge of our game fighters. We began by finding unique 8-bit style characters and adding them to the repository. We created hitboxes that were unique to the characters, added health bars for each opponent, created the attack and jump mechanics, loaded the character's sprite sheets, attached sprites to their respective hitboxes, and have been working on the animations. In terms of the original scope, the project remains on track for completion. However, we are skeptical of additional features as the end of the semester is fast approaching.

4) **Challenges, changes in the plan and scope of the project and things that went wrong during this increment**

   A major challenge we have run into was the correct sizing of the sprite sheet of each character. To better explain, each character in the game has a set number of animations. These animations were combined all into one png to save space. We then will take a square of a constant size and using a loop, iterate over the specific sprite animation to perform the selected action. This works similarly to old movie films, when pictures are shown at high speed it results in movement. However, each sprite animation is of a different pixel size, so the entire png that holds the animations is resized and tested to determine if the animation is correct. This is proving to be very iterative and time-consuming.

5) **Team Member Contribution for this increment**
   - ***a)*** *the **progress report**, including the sections they wrote or contributed to*
   - ***b)*** *the **requirements and design document**, including the sections they wrote or contributed to*
   - ***c)*** *the **implementation and testing document**, including the sections they wrote or contributed to*
   - *d) the **source code** (be detailed about **which** parts of the system each team member contributed to and **how**)*
   - *e) the **video or presentation***

-a

Everyone worked on it together

-b

Everyone worked on it together

-c

Everyone worked on it together

-d

Mason - added the attack hitbox functions, created the sprite sheet of all the animations and coded it into an array in the game

Parker - added health bars for the character. Working on the animations and tweaked the sprite sheet to properly align with the hitbox. I currently am still working on the animation code as it is very tedious.

Parker and Mason also chose two characters samurai and wizard.

-e

Tori

6) Plans for the next increment

- Finalize character special moves and complete the remaining arena designs.
- Begin extensive testing for bugs and game balance.
- Start developing the game's user interface and menu screens.

7) Link to video

https://youtu.be/hxfZjGXW4nI

Software Implementation and Testing Document

For

Group <7>

Version 1.0

Authors:

Mason Russo
Parker Thompson
Tori Lee
Jacob Sonn

Programming Languages (5 points)
Python

Platforms, APIs, Databases, and other technologies used (5 points)
- Python

- Pygame

- GitHub for version control and collaboration

Execution-based Functional Testing (10 points)
Playtests: Conducted multiple rounds of playtesting with team members to evaluate the responsiveness and fluidity of game controls.

Character Control Validation: Each character's control scheme was rigorously tested to ensure that their movements, attacks, and special moves function as intended.

Collision Detection Accuracy: Ensured that the collision detection system accurately registers hits, blocks, and interactions with the game environment.

Game Rule Enforcement: Verified that the game adheres to its established rules, such as scoring, win conditions, and time limits.

Execution-based Non-Functional Testing (10 points)

Load Testing: Stress-tested the game under high load scenarios to ensure it maintains performance without crashing.

Frame Rate Consistency: Monitored the game’s frame rate across different systems to guarantee a smooth gaming experience.

Cross-Platform Compatibility: Tested the game on different operating systems to ensure consistent performance and functionality.

Memory Usage Optimization: Evaluated the game's memory footprint during gameplay to identify and address any memory leaks or excessive usage.

Non-Execution-based Testing (10 points)

Code Reviews: Conducted regular code reviews to maintain code quality, ensure adherence to coding standards, and identify potential bugs.

Documentation Inspection: Reviewed all technical documentation for accuracy and completeness, including comments within the code.

Walkthroughs: Held walkthrough sessions where we explained our code to the rest of the team, fostering understanding and identifying potential issues.

Software Requirements and Design Document

For

Group <7>

Version 1.0

Authors:

Mason Russo
Parker Thompson
Tori Lee
Jacob Sonn

Overview (5 points)

Pixel Pummel" is an innovative 8-bit style street fighter game developed using Python's Pygame library. The game transports players into a pixelated world where they can choose from a variety of unique characters, each with their own special moves and backstories. Set in various thematic arenas, the game offers a competitive fighting experience, blending classic arcade style with modern gameplay mechanics. The goal is to provide a nostalgic yet fresh gaming experience, accessible to both seasoned gamers and newcomers alike.

Functional Requirements (10 points)

1. Character Selection (High Priority): The system shall allow players to select from a roster of eight characters, each with distinctive abilities and aesthetics.
2. Arena Choice (Medium Priority): Players shall be able to choose from four different arenas to play in, with each arena offering a unique visual and strategic environment.

3. Combat Mechanics (High Priority): The system shall enable players to control their characters in combat, including movements like jumping, crouching, and attacking with both basic and special moves.
4. Scoring System (Medium Priority): The game shall keep track of scores based on hits landed and matches won.

Non-functional Requirements (10 points)

1. Performance (High Priority): The game shall maintain a consistent frame rate across all supported platforms.
2. Usability (Medium Priority): The game's controls and interfaces shall be intuitive and easy to learn for players of all skill levels.
3. Compatibility (High Priority): The game shall run smoothly on Windows and MacOS, supporting the latest and two previous major versions of these operating systems.
4. Reliability (High Priority): The game shall not crash or exhibit game-breaking bugs during gameplay.

Use Case Diagram (10 points)

Use Cases Descriptions:

1. Selecting Characters: Players choose their fighters from the character selection screen.
2. Choosing Arenas: Players pick an arena before the start of each match.
3. Engaging in Combat: Players use their characters to fight against an opponent.
4. Viewing Scores: After a match, players see their scores on a summary screen.

Class Diagram and/or Sequence Diagrams (15 points)

Sequence Diagrams:

1. Character Selection Sequence: Details the process from starting the game to selecting a character.
2. Combat Sequence: Showcases the interactions during a typical combat scenario.
3. Scoring Sequence: Describes how scores are calculated and displayed post-match.

Operating Environment (5 points)

Pixel Pummel operates in a PC environment, requiring:

Hardware: Minimum of 4GB RAM and a 2GHz processor.

Operating Systems: Windows 10/11, MacOS Big Sur/Catalina/Monterey.

Software Dependencies: Python 3.8 or higher, Pygame 2.0.1, compatible graphics drivers.

Assumptions and Dependencies (5 points)

Assumptions: Players have basic familiarity with street fighter games; Pygame library will remain supported and updated.

Dependencies: Relies on Pygame for game development; dependent on standard PC input devicess like keyboard.