CIS4951 U01 1258

Dr. Masoud Sadjadi, Florida International University

Capstone II Project

PyQuest

Team Members:

Ana Maria Martinez

Angel Duarte

Jaqueline Lizeth Lopez

Jennifer Mesa

Julian Novak

1. Executive Summary:

PyQuest is a beginner-friendly mobile app game designed to introduce school-age children to the basics of Python programming. Research shows that learning a programming language can improve logical reasoning, strengthen math skills, and foster creativity. The app features a simple login process, level selection, and a structured lesson design to reduce any anxiety when learning to code. This is done through game-like challenges that feature a combat system that prompts the user to solve problems or complete coding challenges to progress.

PyQuest was built in Unity and Visual Studio Code, for smooth visuals, easy navigation, and consistent user experience. With the educational design mixed with the combat-like interaction, PyQuest allows students to engage in problem-solving and build critical thinking skills to help them navigate the world of coding with confidence.

2. Problem & Requirements (O1):

Currently, technology is advancing further and faster, making it crucial for everyone to learn and understand advancements at a similar pace. Despite technological growth, many schools lack the resources to introduce programming concepts that make it engaging for their students. Some schools may be unable to include this aspect of education in their curriculum due to limited funds or time. This neglects a new age of students to possibly create better advancements than the current ones.

To help resolve this problem, PyQuest introduces Python coding concepts while engaging the user with the lessons through game-based learning. The application aims to support the students' learning while also allowing them to move at their own pace. Although this may not

replace a full curriculum, it can serve as an aid or introduction to the concept of coding languages.

Features such as the level selection and progress tracking can help teachers or parents monitor the progress of their students' learning. Ensuring the students make progress at a pace that is suitable for them.

3. System Overview and Architecture (O2):

PyQuest is a 2D game built in C#, using Unity's scene system to control the players movement through different parts of the game. The app is structured using multiple Unity scenes such as the Login, Main Menu, Level selection, and the levels themselves. The level layouts are created using Tiled and imported into Unity using the Tiled2Unity and SuperTiled2Unity workflow. The C# script handles the character movement, animations, and player interactions.

The architecture flow looks like this:

- Login
- Menu
- Level selection
- User Gameplay

The data is stored in the simple app state (username + level progress)

4. Discipline-Specific Depth (O6):

The application incorporates an educational flow algorithm that promotes mastery-based learning. The algorithm keeps the student within a topic until they have shown full mastery before advancing to the next lesson. There is a quiz built in after each lesson to show mastery or current understanding of the concepts. By locking the lessons until mastery is achieved, the system keeps the students focused on the lesson one at a time, which prevents any gaps that could arise. This algorithm reinforces critical thinking and conceptual clarity.

5. Implementation Notes (O2):

PyQuest is implemented with Unity and built with C# scripts written in Visual Studio Code. The team used Unity's 2D tools for all movements, collisions, and animations. The sprite animations are controlled using Unity Animator, and transitions depend on the player's input/movement. The code format is organized by feature:

- Player movement script
- Chest interaction script
- Quiz system script
- Level loading script

Each scene has its own GameObjects and scripts to keep the project simple and clear. The logic runs locally in Unity; no external server or database is needed. The map layouts were imported into Unity and used for the level designs, to avoid creating each tile one by one.

6. Testing and Evaluation (O3):

Testing for PyQuest focused on clarity, validating usability, reliability, and verifying the interface with the educational flow algorithm. Testing involved manual UI testing where users would navigate through the app, enter information, attempt differing actions, and describe any hard-to-understand prompts.

Along with the UI testing, user feedback was gathered from school-aged children, family members, friends, and colleagues. These users help provide insight into how the instructions, lesson explanations, and quiz prompts were easy to comprehend. This helped the team fix any unclear language, issues regarding inputs, and navigation issues. Based on the results of the testing, the team document the user's difficulties, fix any bugs, suggest new features/improvements, and test once more. Priority would lie with ensuring that the lessons were easy to comprehend and the quizzes were simple to navigate through.

7. Security, Privacy, Accessibility (O5):

PyQuest was designed with attention to the safety and usability needs of young learners. To ensure the protection of the users, user data was not stored externally. Rather, data is stored temporarily in the in-app variables. The app does not collect personal data and does not rely on external networks or API's. For accessibility, the interface uses large fonts, high-contrast colors, and age-appropriate language. These choices were meant to keep the students engaged and reduce the cognitive load.

8. Deployment & Operations:

PyQuest runs by opening Unity and pressing the play button, with no backend deployment necessary. All the features and developments were conducted on Unity with Visual Studio code; making it easy to test and demo the project with no additional configuration.

Demonstrating the application:

- Launch Unity with Visual Studio code
- Open the PyQuest
- Run the app
- Interact with the lessons, quizzes, and navigation

Future deployment of, sharing, and testing of the app involves exporting a build for Windows, Mac, or mobile later if needed.

9. Limitations and Future Work:

There are several limitations within the application, which the team plans to expand in the future. Some of these limitations involve the app only supporting a few lessons, having no backend storage, and the animations are simplistic in nature. For the lessons, the team will work on future versions that will expand the curriculum and versions that will save long-term progress saving without losing the player's progress. Other future goals involve adding animations, improving character movement, and making the app accessible on other platforms. The team aims to make the app more interactive and interesting by adding more puzzles, levels, and boss battles.

10. References:

[1] Unity Technologies, “Unity 2D Game Development Documentation,” Unity, 2024.

[2] Microsoft, “C# Programming Guide,” Microsoft Learn, 2024.

[3] S. Y. Lye and J. H. L. Koh, “Review of research on game-based learning in computer science education,” *Computers & Education*, vol. 79, pp. 353–364, 2014.

[4] M. Papastergiou, “Digital game-based learning in high school computer science education: Impact on student engagement,” *Computers & Education*, vol. 52, no. 1, pp. 1–12, 2009.

[5] Code.org, “The importance of computer science education,” Code.org, 2023.

[6] Python Software Foundation, “Python for Beginners,” python.org, 2023.

[7] SuperTiled2Unity Documentation, “Importing Tiled Maps into Unity,” 2024.