GROUP ASSIGNMENT

TECHNOLOGY PARK MALAYSIA

CT029-3-2-ISE

IMAGING AND SPECIAL EFFECTS

INTAKE: APU2F2411CS(AI)

DATE ASSIGNED: 16th JUNE 2025

DATE DUE: 12th AUGUST 2025

GROUP NUMBER: 42

LECTURER: DR. VAZEERUDEEN ABDUL HAMEED

No.	Student Name	TP Number
1.	MANREEN KAUR A/P JAGJIT SINGH	TP071290
2.	SEHBA HANIEF	TP078886
3.	HASIBUL ISLAM	TP077662
4.	PANOV EGOR	TP077925

Table of Contents

Abstract

This project is a simple 2D side-scrolling game called Justin Dash for Love, developed using Python and the Pygame library. The game is designed for PC and focuses on an endless running mechanic where the player controls the character Justin, who must jump over obstacles to progress. The goal is to survive as long as possible while maintaining a high score. It targets casual gamers who enjoy fast-paced reflex-based games. The main objective is to combine fun gameplay with simple mechanics and engaging visual effects. This project demonstrates knowledge of Python programming, Sprite animation, collision detection, and basic game physics. The game also incorporates particle effects for added visual feedback, making it more interactive and enjoyable for the player.

Introduction

The game Justin Dash for Love is a fast-paced 2D platformer where the main character, Justin, goes on an adventurous journey to reach his loved one. The story begins in a small and peaceful town, where Justin's girlfriend is taken away by an unknown enemy. Determined to save her, Justin sets out across multiple levels filled with obstacles, enemies, and traps. The player controls Justin as he runs, jumps, and dashes through each stage, avoiding hazards while collecting items along the way. The narrative is simple but engaging at each level brings Justin closer to his goal, and the challenge increases as new dangers appear. The game aims to combine fun gameplay mechanics with an easy-to-understand storyline so that players of all ages can enjoy the experience. The setting and characters are inspired by classic retro games but enhanced with modern animations, colorful environments, and smooth controls.

Project Objectives

1. Build a playable 2D side-scroller/runner with two full levels (Level 1: 50 m; Level 2: 100 m total).
2. Create original visual assets characters, obstacles, and backgrounds using Photoshop that we drew in soft texture and GIMP.
3. Implement smooth player controls and lane movement using keyboard input (arrow keys or A/D for left-right; Space/Up to jump).
4. Add a scoring system where players collect hearts and power-ups, and implement a basic health/life system.
5. Integrate audio with background music for each level and SFX for jump, collision, and pickups.
6. Implement visual special effects such as particle bursts, hit flash, and parallax background scrolling.
7. Use event-driven programming (Pygame) for input handling, collision detection, powerup timers, and level transitions.
8. Test gameplay mechanics and fix issues so the game runs reliably at 60 FPS on a standard PC.

Scope of Project

The scope of this project covers the design and development of a single-player endless running game for desktop systems. The game runs in a fixed resolution and is intended for casual entertainment. It includes core mechanics such as running, jumping, collision detection, and scoring. Special effects are added to improve player feedback. The project does not include online multiplayer features, advanced AI, or level progression beyond endless running. The scope also includes the creation of game assets, sound effects, and testing for smooth performance on mid-range computers. The main limitation is that the game is built for PC only and may require manual installation of Python and Pygame to run.

Tools and Technologies Used

The game Justin Dash for Love was developed using **Python** with the **Pygame** library as the main framework. Pygame was chosen because it supports sprite rendering, animation, event handling, and audio playback, all of which are essential for a 2D platformer. It is also lightweight, beginner-friendly, and well-documented, making it ideal for this project.

The code is organised into separate modules such as main.py, asset_loader.py, player.py, level_manager.py, particle_manager.py, and playing_state.py to keep the logic modular and easier to maintain.

For visual assets, **Photoshop** was used to design and make them in soft texture and model the characters and objects, while **GIMP** was used to clean, resize, and export sprites and layered backgrounds for parallax scrolling. The backgrounds include assets/images/backgrounds/level1_park.png for the first level and level2_industrial.png for the second level.

For audio, **Audacity** was used to trim, balance, and loop short music tracks and sound effects. Additional sound effects and music were sourced from royalty-free libraries and optimised to work efficiently within Pygame's rendering system.

Development and testing were done on a standard desktop computer using **Visual Studio Code** as the code editor because of its debugging tools and extension support. Game settings such as SCREEN_WIDTH = 800, SCREEN_HEIGHT = 600, and FPS = 60 are defined in settings.py to ensure smooth performance.

These tools and technologies were chosen because they are reliable, freely available, and well-suited for modelling, image editing, audio processing, and 2D game programming.

Storyboard Narrative

The game begins with an opening cutscene showing Justin at his home. Suddenly, the antagonist appears and kidnaps Justin's girlfriend, setting the story in motion. The scene then fades into the first playable level, showing Justin standing at the starting point, ready to begin his rescue mission.

In Scene 1, the player controls Justin as he runs through a peaceful grassy environment filled with small obstacles and collectible coins. Birds can be seen flying in the background, adding life and movement to the scene. The music is light and upbeat, reflecting the relatively easy start to the adventure. This level is designed to introduce the player to the game's mechanics, such as running, jumping, and avoiding basic hazards.

In Scene 2, the environment changes drastically to a darker, more dangerous castle setting. The ground is made of stone, and the background features looming walls and dim torches. Pits, moving platforms, and patrolling enemies make navigation more challenging, requiring careful timing and precision. The music becomes more intense to match the heightened danger. As Justin advances, he eventually reaches the antagonist for the final confrontation. Upon defeating the enemy, a short ending scene plays, showing Justin successfully rescuing his girlfriend and celebrating his victory.

Transitions between scenes are smooth, giving a sense of continuous storytelling while gradually increasing the difficulty. Each scene is designed to naturally guide the player through the narrative while keeping the gameplay engaging.

Screenshots Of The Game Scene and Explain The Assets

1. Main Menu Screen

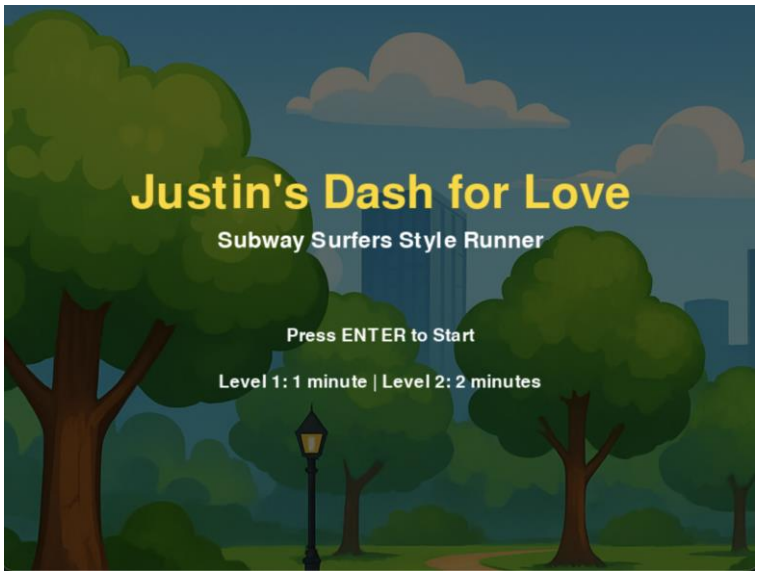

Figure 1 Main Menu Screen

- Displays the title Justin Dash for Love in bold, colorful letters at the top.
- Buttons for Start, Settings, and Exit are centered, with hover effects for user feedback.
- Uses the Level 1 parallax background and soft, looping background music to create a welcoming atmosphere.

2. Level 1 Gameplay Scene (Park Theme)

- Player Character: Justin, with animations for idle, running, jumping, and falling.

Figure 2 Player Character

- Background: Parallax scrolling (level1_park.png) with distant mountains, mid-ground trees, and foreground grass.

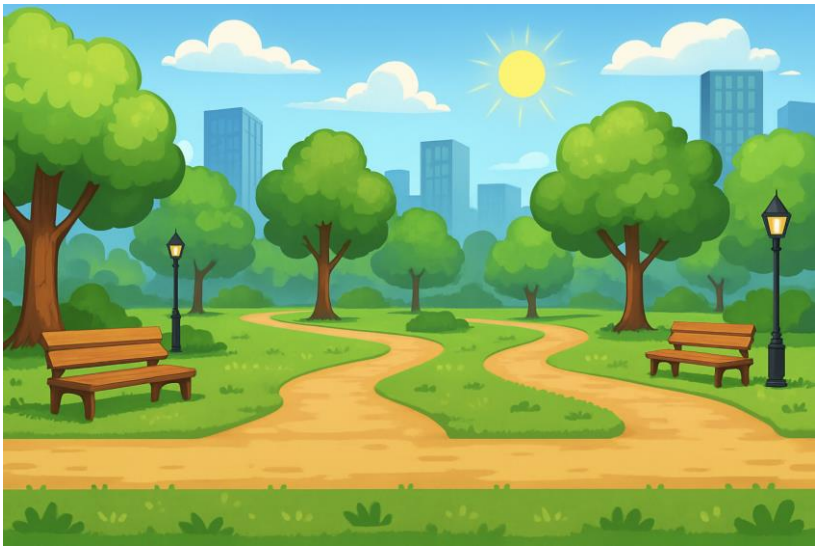

Figure 3 Background Level 1

- Platforms: Tile-based for consistent collision detection.
- HUD: Displays score, health/lives, and elapsed time.

Figure 4 Hud Level 1

- Obstacles and Enemies: Metallic spikes and moving platforms. Patrolling creatures with walk-cycle animations and simple AI.

Figure 5 Obstacles and Enemies Level 1

Figure 6 Obstacles and Enemies Level 1

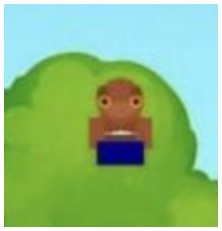

Figure 7 Obstacles and Enemies Level 1

- Collectibles: Bright gold coins (score).

Figure 8 Collectibles

Figure 9 Collectibles

Figure 10 Collectibles

- Effects: Particle animations when collecting or dashing.

3. Level 2 Gameplay Scene (Industrial Theme)
    - Background: level2_industrial.png with factories, smokestacks, and darker tones.

Figure 11 Background Level 2

    - Platforms: Metal and stone textures to match the theme.
    - Enemies & Obstacles: More aggressive enemies, faster-moving platforms, and complex spike arrangements. It is the same as level 1 enemies and obstacles.
    - Collectibles: Bright gold coins (score). It is the same as the level 1 collectibles score.
    - Music: Faster-paced soundtrack for higher tension.
    - HUD: Same elements as Level 1, updated in real-time.

Figure 12 Hud Level 2

4. Game Over Screen
    - Displays “Game Over” message with Retry and Return to Menu buttons.

Figure 13 Game Over Screen

    - Background music switches to a slow, somber tune.
5. Victory Screen

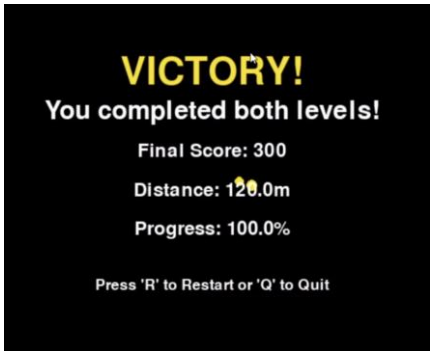

Figure 14 Game Over Screen

- Celebratory screen with congratulatory message for rescuing Justin's girlfriend.
- Includes particle effects like confetti bursts and upbeat victory music.
- Buttons for Play Again and Main Menu enable replay.

Game Design & Development Process

The development process started with planning the concept, defining the mechanics, and gathering required assets. The next step was setting up the main game loop in Python, where events, updates, and rendering occur every frame. The Player class (player.py) manages movement, animations, and collision detection. The Level Manager handles obstacle creation and background scrolling. The Particle Manager (particle_manager.py) adds dust and impact effects at specific events. Testing was done continuously during development to ensure mechanics worked as expected. The game was finalized by adding menus, scoring, and final polish.

Special Effects & Implementation

Visual special effects are handled mainly by the **particle_manager.py** file. When Justin jumps, a dust particle animation plays near his feet, simulating a burst of air. When a collision occurs, particles scatter from the impact point, giving instant visual feedback. These effects are created using small PNG sprites with alpha transparency, updated each frame to simulate motion and fading.

Audio effects are integrated using Pygame's mixer module. A jump sound plays each time the player presses space, and a collision sound triggers when hitting an obstacle. These sounds were chosen to match the game's playful and lighthearted tone, enhancing immersion. The combination of visuals and audio creates a more dynamic and responsive feel for the player.

Testing & Evaluation

Testing was conducted throughout development to check player movement, jumping, collision accuracy, scoring, and menu navigation. Different scenarios were tested, including rapid jumps, multiple obstacles, and high-speed gameplay. Bugs such as inaccurate collision detection and particles not disappearing were fixed. The game runs smoothly at 60 frames per second, with no major performance drops. Player feedback confirmed that controls are responsive, and visuals are engaging.

Challenges / Solutions / Weaknesses

The game does have some weaknesses that could impact long-term enjoyment. Its overall length is relatively short, with only two levels, which may lead experienced players to complete it quickly. Enemy variety is limited, making encounters feel somewhat predictable over time. There is minimal incentive for replay, as the game lacks unlockables, achievements, or alternate endings that typically encourage repeated playthroughs. The difficulty jumps between levels, particularly from the first to the second, can feel steep and potentially frustrating for some players. The absence of a saves or checkpoint system means that failure near the end of a level forces a full restart, which could discourage casual players. Player abilities are also limited to basic actions such as running, jumping, and dashing, with no upgrades or special moves to evolve the gameplay. Lastly, the game is single-player only and does not offer multiplayer or cooperative modes, limiting replayability for those who enjoy shared gaming experiences.

Throughout development, the team encountered several challenges and devised effective solutions. One key challenge involved collision detection accuracy; oversized hitboxes initially caused players to lose unfairly. This was resolved by adjusting hitbox sizes to better match the sprite images, resulting in more precise collision handling. Maintaining particle effects' visual appeal without sacrificing performance was another hurdle. The team implemented object pooling, which allowed for reusing particle objects instead of creating new ones each frame, improving efficiency. Asset compatibility issues also arose, with inconsistencies in image size and format leading to loading errors. To fix this, all assets were converted to compatible formats and standardized in dimensions. Ensuring smooth level transitions posed a challenge as well, as scene switching needed to be instant and lag-free. Preloading level assets and optimizing memory management helped achieve seamless transitions. Audio balancing was another concern, as music and sound effects sometimes clashed in volume; this was addressed by fine-tuning audio levels and testing on multiple systems. Finally, synchronization between moving platforms and the player was tricky, causing occasional clipping or unexpected falls. Enhancing platform physics improved synchronization and prevented these issues.

Conclusion

Justin Dash for Love is a fully developed 2D endless runner game for PC, created using Python and Pygame. It features automatic running, jump controls, precise collision detection, scoring, particle effects, and simple menus all working together to deliver a smooth and visually appealing gameplay experience. Throughout development, challenges such as fine-tuning collision detection, optimizing particle effects, and ensuring asset compatibility were successfully addressed, resulting in a polished and stable final product.

The game's key strengths lie in its simple and intuitive controls, making it accessible to players of all ages, including casual gamers. Smooth and fluid animations for the main character, enemies, and particle effects contribute to a polished presentation. Performance remains consistently stable, running at 60 frames per second without noticeable lag, even on mid-range computers. The visual appeal is enhanced by engaging particle effects, parallax scrolling backgrounds, and vibrant collectibles. Complementing this is atmospheric audio with background music tailored to each level, creating an immersive environment. Players also benefit from variety through two distinct environments a grassy park and an industrial factory each offering unique aesthetics and challenges. The straightforward narrative provides clear motivation that keeps players engaged and progressing.

Looking ahead, future enhancements could further enrich the gameplay experience. Adding more diverse environments would extend playtime and keep the game feeling fresh. Introducing additional enemy types with varied behaviors would increase challenge and player engagement. Implementing a power-up system could add strategic depth, while cutscenes between levels might help develop the narrative and deepen player connection. Overall, Justin Dash for Love offers a strong foundation with ample room for expansion to create an even more captivating and enjoyable game.

References

1. (2020). Inventwithpython.com. https://inventwithpython.com/pygame/chapter2.html
2. Pygame Front Page — pygame v2.6.0 documentation. (2023). Pygame.org. https://www.pygame.org/docs
3. Fincher, J. (2019, September 16). PyGame: A Primer on Game Programming in Python. Realpython.com; Real Python. https://realpython.com/pygame-a-primer/#sprite-images
4. Tech With Tim. (2017, November 7). Pygame Tutorial #1 - Basic Movement and Key Presses. YouTube. https://www.youtube.com/watch?v=i6xMBig-pP4&list=PLzMcBGfZo4-lp3jAExUCewBfMx3UZFkh5&index=1
5. Tech With Tim. (2018, February 6). Pygame Tutorial #3 - Character Animation & Sprites. YouTube. https://www.youtube.com/watch?v=UdsNBIzsmlI&list=PLzMcBGfZo4-lp3jAExUCewBfMx3UZFkh5&index=3
6. Tech With Tim. (2018, February 21). Pygame Tutorial #6 - Enemies. YouTube. https://www.youtube.com/watch?v=vc1pJ8XdZa0&list=PLzMcBGfZo4-lp3jAExUCewBfMx3UZFkh5&index=6
7. Tech With Tim. (2018, February 23). Pygame Tutorial #7 - Collision and Hit Boxes. YouTube. https://www.youtube.com/watch?v=1aGuhUFwvXA&list=PLzMcBGfZo4-lp3jAExUCewBfMx3UZFkh5&index=7
8. Tech With Tim. (2018, March 2). Pygame Tutorial #8 - Scoring and Health Bars. YouTube. https://www.youtube.com/watch?v=JLUqOmE9veI&list=PLzMcBGfZo4-lp3jAExUCewBfMx3UZFkh5&index=8
9. Tech With Tim. (2018, March 8). Pygame Tutorial #9 - Sound Effects, Music & More Collision. YouTube. https://www.youtube.com/watch?v=2BikxsbkuIU&list=PLzMcBGfZo4-lp3jAExUCewBfMx3UZFkh5&index=9

10. Craven, P. V. (2017). Program Arcade Games With Python And Pygame. Programarcadegames.com. http://programarcadegames.com/index.php?lang=en&chapter=introduction_to_animation
11. Programming MOOC 2022. (2025). Mooc.fi; University of Helsinki. https://programming-25.mooc.fi/part-13/2-animation
12. GettingStarted - pygame wiki. (n.d.). Www.pygame.org. https://www.pygame.org/wiki/GettingStarted

Workload Distribution

Task Description	Member 1 (Manreen Kaur)	Member 2 (Sehba Hanief)	Member 3 (Hasibul Islam)	Member 4 (Panov Egor)
Game concept & story writing	25%	25%	25%	25%
Game asset design (characters, backgrounds, UI)	25%	25%	25%	25%
Programming (game logic, controls, collisions)	25%	25%	25%	25%
Special effects integration (visual & audio)	25%	25%	25%	25%
Testing & debugging	25%	25%	25%	25%
Documentation writing & formatting	25%	25%	25%	25%
Final presentation preparation	25%	25%	25%	25%