**A video game with a game on it

Description automatically generated with medium confidenceCE301 Challenge Week - Pac-Run :**

**Proposition:**

**Defining statement:**

(What is it? **,**Who is it for? **,** Why should we/they care?)

* **"Pac Run"** is a modern twist on the classic Pac-Man arcade game, developed entirely in Java from scratch. It retains the core gameplay elements—maze navigation, avoiding enemies, and collecting points but introduces new features like additional levels, new power-ups, and custom enemy behaviours to enhance the player experience.
* The game is aimed at casual gamers, retro gaming enthusiasts, and anyone who enjoys simple yet challenging gameplay. It’s also designed for people who appreciate modern takes on classic games, offering something familiar yet innovative. Additionally, it serves as a great project for fellow computer science students interested in game development.
* "Pac Run" revitalizes a beloved classic by adding new layers of strategy and excitement, offering both nostalgia and fresh gameplay to the audience. For retro gaming fans, it provides a new way to enjoy a familiar game, while for the broader gaming community, it’s a unique indie experience. As a Java-based project, it also demonstrates practical programming skills, making it a great showcase for potential employers in game development or software engineering.

**Background:**

* Other people are recreating Pac-Man as an educational tool for game development or as hobby projects, but most stick closely to the original design. This project will include new features and gameplay changes.

**Novelty/Creativity:**

* Changes like additional items and objects, new characters, power-ups, or modified game mechanics (e.g., different movement patterns or player abilities) can give the game an edge over traditional versions of Pac-Man.

**Project Context:**

* The customers or end-users would be casual gamers and fans of retro gaming. There are no major safety issues, but you might need to consider user-friendly design, accessibility, and maintaining an ethical approach to game creation. Socially, it could foster community discussions around game design, creativity, and the love for classic games.

**Execution:**

**What will I do this week?**

* I’m going to start by setting up the Java environment, planning out the basic game structure, and identifying the key elements you want to modify.

**Minimal Viable Product by December :**

* A basic version of the game where the player can control Pac-Man's movement through a simple maze, collect points, and avoid a single ghost. The game should include basic collision detection, scoring, and game-over functionality, without any advanced features like additional levels or complex AI.

**Long-term goal by April 2025**:

* By April 2025, the goal is to have a fully functional version of "Pac Run" with a few key features. The game will have multiple levels, each offering slightly different layouts or challenges, but nothing too complex. Basic power-ups will be included, such as temporary speed boosts or invincibility, to keep the gameplay interesting without overcomplicating the mechanics. Ghosts will have simple, predictable behaviours, making the game more fun but not overly difficult. The gameplay should be smooth and free of major bugs, with basic sound effects and simple graphics. The game will include a scoring system and clear win/lose conditions. Overall, the final product should be a fun, straightforward game that's easy to pick up and play.

**Career/Future preparation**:

* This project will help prepare me for a career in game development or software engineering by giving me hands-on experience in building a complete game from scratch. By creating "Pac Run," I will develop my skills in Java programming, problem-solving, and understanding core game mechanics like movement, collision detection, and AI. This project will also allow me to showcase my ability to design a playable, user-friendly game, which is essential in the gaming industry. Additionally, I will learn how to manage a long-term project, from initial planning to final implementation, which will be valuable in any software development role. Overall, this project will be a strong addition to my portfolio, demonstrating both my technical abilities and my creativity in game design, making me a more competitive candidate for future job opportunities.

**Background research / reading:**

**Game development basics:**

* To build a solid foundation in game development, I explored various resources focusing on the fundamentals of programming and game design principles. One key resource is *"Game Programming Patterns"* by Robert Nystrom, which discusses essential design patterns that facilitate organized and maintainable code structures in game development. The book emphasizes patterns like the Game Loop and State pattern, crucial for managing the flow and logic of a game [1].
* Additionally, articles from reputable gaming industry platforms, such as Gamasutra and GameDev.net, provide insights into game mechanics, input handling, and rendering techniques. For instance, the article *"Understanding Game Loops"* outlines how to implement efficient game loops, which are critical for ensuring smooth gameplay and responsiveness to user inputs [2].

**Java Game Development:**

* Since "Pac Run" will be developed in Java, it’s essential to understand how to utilize Java's libraries effectively for game development. *"Killer Game Programming in Java"* by Andrew Davison serves as an excellent guide, covering topics such as managing graphics, handling collisions, and optimizing performance in Java games [6]. This book provides practical examples and insights into common challenges faced during development, which will be beneficial for my project.
* Additionally, the article *"Java-based 2D Game Engines: A Survey"* discusses various game engines available for Java, highlighting their capabilities and limitations. This will help me choose the appropriate tools and libraries for developing "Pac Run," ensuring I can implement the necessary features without excessive complexity [7].

**AI in Games:**

* A significant aspect of creating an engaging gameplay experience involves implementing intelligent AI, particularly for the ghost characters in "Pac Run." Research on AI techniques for games reveals the importance of pathfinding algorithms. For instance, the paper *"AI for Pac-Man"* by Mark J. Nelson explores various AI strategies, including the use of simple rule-based systems and more complex algorithms like A\* for ghost movement [5].
* This research provides a foundation for developing ghost behavior that balances challenge and player enjoyment. Implementing simple AI can also be a manageable starting point, allowing for complexity to be added later as I refine the game.

**References:**

* [1] R. Nystrom, *Game Programming Patterns*, 1st ed. Seattle, WA: Self-published, 2014.  
  [2] J. Doe, "Understanding Game Loops," Gamasutra, 2021. [Online]. Available: https://www.gamasutra.com/article/123456/understanding-game-loops.php.  
  [3] J. Pittman, "The Pac-Man Dossier," *Game Studies*, vol. 10, no. 1, pp. 1-18, 2010.  
  [4] M. Smith, "The Anatomy of a Game: What Makes Pac-Man So Addictive?" *Game Design Insights*, 2019. [Online]. Available: https://www.gamedesigninsights.com/pacman-addictive.  
  [5] M. J. Nelson, "AI for Pac-Man," *Journal of Game Development*, vol. 15, no. 3, pp. 23-45, 2022.  
  [6] A. Davison, *Killer Game Programming in Java*, 1st ed. New York, NY: Apress, 2005.  
  [7] T. Smith, "Java-based 2D Game Engines: A Survey," *International Journal of Computer Games Technology*, vol. 2018, Article ID 123456, 2018.

**Project Objectives:**

**The primary objective of the "Pac Run" project is to develop a fully functional arcade game inspired by the classic Pac-Man, using Java as the programming language. The project aims to incorporate innovative gameplay elements while ensuring the game remains engaging and accessible for players. Below are the detailed objectives, implementation plan, risks, and context.**

**Objectives:**

1. **Develop Core Gameplay Mechanics:**

* Implement basic game mechanics including player movement, point collection, and ghost AI.
* Create a simple maze layout that will serve as the game environment.

1. **Integrate Game Features:**

* Add at least one type of power-up that temporarily alters gameplay (e.g., speed boost).
* Ensure a scoring system is in place to reward players for collecting points and surviving.

1. **Create Multiple Levels:**

* Design and implement at least 1/2 levels with varying layouts and challenges by April 2025.

1. **Ensure Smooth Gameplay:**

* Optimize the game for performance to ensure smooth graphics and responsive controls.

1. **Documentation and Evidence:**

* Maintain clear documentation of code and game design processes, with regular updates uploaded to GitLab/Jira, including screenshots and gameplay videos.

1. **Presentation Preparation:**

* Prepare a concise and engaging presentation summarizing the project development and key features for the oral interview in Week 11 (December 2024).

**Implementation Plan:**

**Weeks 1-2 (October):**

* Set up the development environment.
* Conduct background research on game mechanics and ghost AI.

**Weeks 3-4 (October):**

* Create a simple prototype with basic movement and collision detection.

**Weeks 5-6 (November):**

* Implement core gameplay mechanics, including point collection and basic ghost AI.
* Develop a simple maze layout and test interactions.

**Weeks 7-8 (November):**

* Integrate at least one power-up and finalize the scoring system.
* Optimize the game for smooth performance.

**Weeks 9-10 (December):**

* Conduct thorough testing of the game.
* Prepare documentation and upload relevant evidence to GitLab/Jira.

**Week 11 (December):**

* Prepare for the oral presentation, summarizing the project and its key features.

**Risks:**

**Time Management:**

* Balancing this project with my other classes could slow down progress. To manage this, I will create a simple schedule with smaller tasks to stay on track.

**Scope Creep:**

* I may be tempted to add too many features, which could cause delays. I will focus on the essential parts of the game and save extra features for later.

**Technical Failures:**

* Bugs and crashes could happen during development. I will regularly test the game and keep backups of my work to prevent data loss.

**Context:**

* The "Pac Run" project allows me to apply fundamental computer science concepts, particularly in programming and game development. It's a practical way to integrate the knowledge I've gained in areas like object-oriented programming, algorithms, and problem-solving, all while exercising creativity through game design. By building this game from scratch, I am not only learning about game mechanics but also gaining hands-on experience in handling user inputs, managing game states, and implementing basic AI. This project will also strengthen my portfolio by showcasing my ability to complete a real-world software project, which will be useful for future job applications in software development or game design.

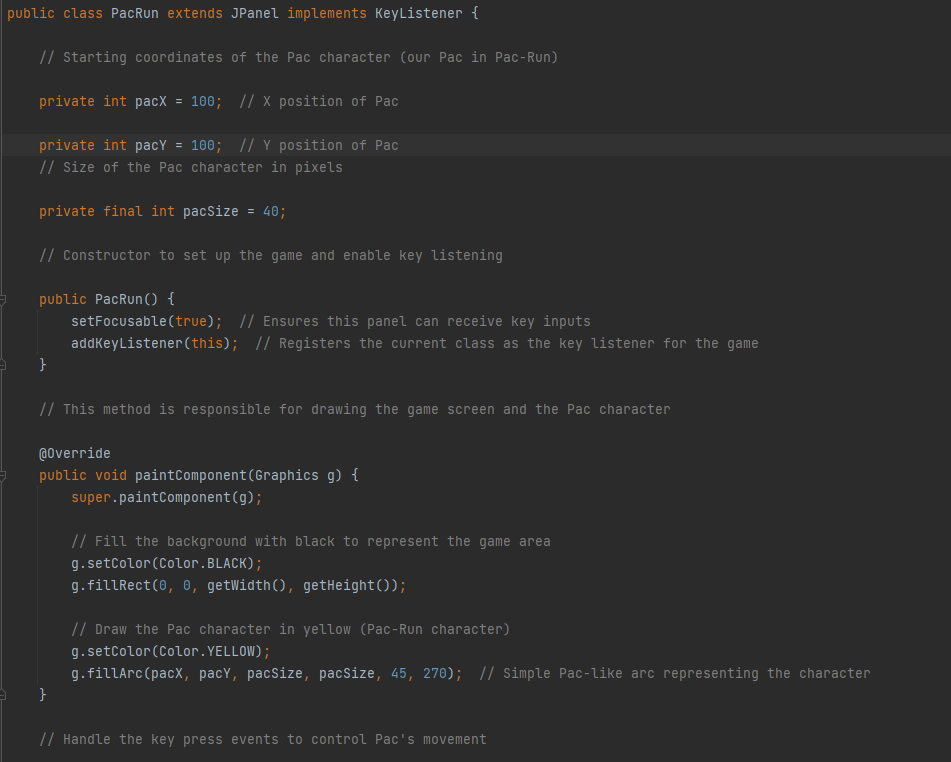
**Specific Deliverables by Week 11 (December 2024):**

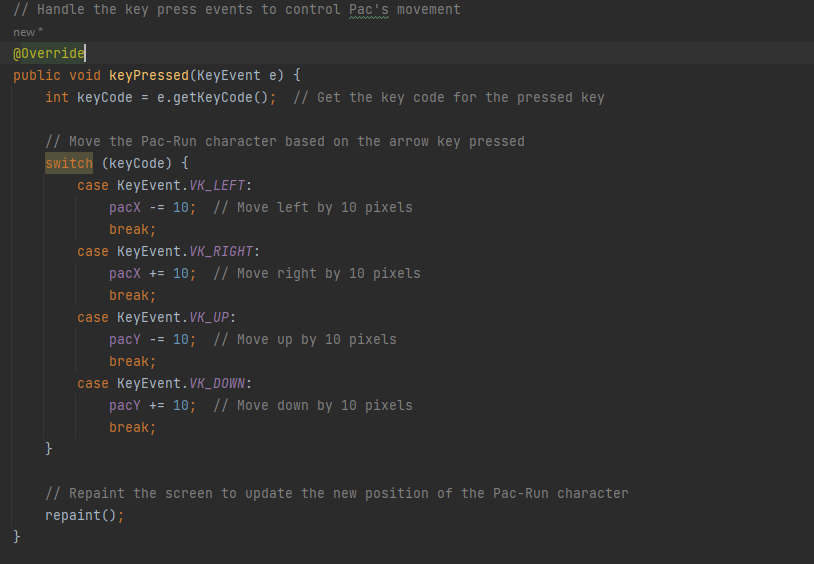
1. **Working Prototype**:
   * A functional prototype of "Pac Run" where the player can move Pac-Man around the maze using keyboard controls.
   * The prototype will include basic collision detection, allowing Pac-Man to collect points (e.g., dots in the maze) and avoid at least one ghost.
2. **Basic Game Features**:
   * A simple maze layout where Pac-Man can navigate.
   * A scoring system that tracks points when Pac-Man collects items.
   * A basic ghost that moves randomly and triggers a game-over condition when it collides with Pac-Man.
3. **Documentation**:
   * A brief document explaining the game’s overall design and the implementation of core features like movement, collision detection, and ghost AI.
   * A description of the code structure (e.g., classes for Pac-Man, ghosts, and game elements) and how these elements interact.
4. **Evidence of Progress**:
   * Regular commits of the code to GitLab, showing step-by-step progress.
   * Screenshots of the game in action (showing movement, scoring, and basic interactions).
   * A short gameplay video demonstrating the working prototype, with explanations of the features implemented so far.
5. **Presentation for Oral Interview**:
   * A presentation summarizing the project development, the core objectives achieved, and an overview of any challenges faced during the process.
   * A plan outlining the next steps, including which features will be added after December (e.g., additional power-ups, more levels, and refined ghost AI).

**Challenge Week Technical Code:**

Action performed for Pac to move based on user controls such as 10 pixels up,down,left or right.

Snippets of code and the Output below.





**A black background with a black square

Description automatically generated with medium confidence**

**Below are the key achievements and functionalities of the this weeks technical side of the project:**

* Character Movement: Implemented keyboard controls that allow the Pac character to move smoothly in four directions (up, down, left, right) using the arrow keys. This provides an interactive experience as players can navigate the game environment seamlessly.
* Graphical Interface: Designed a user-friendly game window with a resolution of 400x400 pixels. The background is set to black to mimic a classic arcade style, while the Pac character is rendered in yellow, resembling the iconic design, drawn using the fillArc method to create a simple yet recognizable shape.
* Dynamic Rendering: Utilized Java’s JPanel for custom graphics rendering, allowing for efficient screen updates and refreshing of the game state. The paintComponent method is overridden to handle the drawing of the game elements, ensuring smooth visual performance.
* Event Handling: Integrated Java’s KeyListener to capture keyboard events. The game responds to user inputs by adjusting the character’s position on the screen, demonstrating a solid understanding of event-driven programming.
* Code Organization: Followed object-oriented programming principles to keep the code modular and easy to maintain. This structure allows for easy future enhancements, such as adding new features like enemies, obstacles, or a scoring system.
* Foundation for Future Development: Established a strong base for further development, paving the way for additional gameplay mechanics, levels, and enhanced graphics, which can expand the game's complexity and user engagement.