Your game concept for "Armor" sounds like a compelling blend of RPG and shooter elements with a clear, engaging objective. Given the details, using Pygame for initial development makes sense due to its simplicity and effectiveness for 2D games. PyOpenGL can be considered for more advanced graphical features or performance optimizations later in the development process. However, targeting mobile platforms with Python, especially with Pygame, can be challenging because Pygame is primarily designed for desktop applications. For a mobile release, you might need to look into tools like Kivy, which is more suited for mobile app development in Python, or consider porting the game to a mobile-friendly engine later on.

### Development Plan Overview:

1. \*\*Game Design\*\*:

- Flesh out the storyline, levels, and enemy types.

- Design the upgrade system for the player's armor and weapons.

- Create simple sprite sheets for the player, enemies, and environment.

- Collect or create MIDI sounds and DOOM-like SFX for the game's audio.

2. \*\*Development Tools\*\*:

- \*\*Pygame\*\* for creating the game window, handling events, rendering sprites, and playing sounds.

- \*\*PyOpenGL\*\* for advanced graphics features if needed (optional at the start).

- Consideration of \*\*Kivy\*\* or other frameworks for the mobile version, or exploring tools for porting the game to mobile platforms.

3. \*\*Core Game Components\*\*:

- \*\*Game Initialization\*\*: Set up the game window, load sprites, and initialize game variables.

- \*\*Main Game Loop\*\*: Implement the loop to handle events (e.g., player input), update game states (e.g., player and enemy positions), and render the updated game world each frame.

- \*\*Player Mechanics\*\*: Code for player movement, shooting, and armor/weapon upgrades.

- \*\*Enemy Mechanics\*\*: Define behaviors for different enemies, including movement patterns and attack methods.

- \*\*Level Progression\*\*: Mechanism to clear levels, transition to new levels, and save game progress.

- \*\*UI/UX\*\*: Design and implement user interfaces for game status, inventory management, and upgrade systems.

4. \*\*Prototype & Iteration\*\*:

- Start by creating a basic prototype focusing on player movement, shooting mechanics, and simple enemy interactions.

- Gradually add more features, such as armor upgrades, more complex enemy types, and level progression.

5. \*\*Polishing & Testing\*\*:

- Refine game mechanics based on playtesting feedback.

- Optimize performance, particularly if planning to use PyOpenGL for enhanced graphics.

- Ensure compatibility with desired platforms, focusing initially on desktop.

### Starting with Pygame:

Given the scope of your project, let's sketch out an initial setup for your game using Pygame. This will include initializing the game window, setting up the main game loop, and handling basic player input. Note that this is a high-level overview; each part of the game will require more detailed development.

Would you like to proceed with a detailed code example for setting up the game window and handling basic player movements, or do you have specific aspects of the game development process you'd like to focus on first?

**90 DAY DEVELOPMENT PLAN**

**### 90-Day Development Plan for "Armor"**

**#### \*\*Phase 1: Planning and Initial Setup (Days 1-15)\*\***

**\*\*Goals\*\*:**

**- Define game mechanics and overall design**

**- Set up the development environment**

**- Create basic game structure**

**\*\*Tasks\*\*:**

**1. \*\*Game Design Document\*\*:**

**- Outline the game’s core mechanics, story, levels, and upgrade system.**

**- Design player and enemy sprites, environmental elements, and UI components.**

**2. \*\*Development Environment Setup\*\*:**

**- Install Python, Pygame, and any other necessary tools.**

**- Set up version control (e.g., GitHub) for the project.**

**3. \*\*Basic Game Structure\*\*:**

**- Initialize a Pygame window.**

**- Create a main game loop to handle events and rendering.**

**\*\*Deliverables\*\*:**

**- Completed game design document**

**- Basic Pygame window with a main game loop**

**#### \*\*Phase 2: Core Mechanics Implementation (Days 16-45)\*\***

**\*\*Goals\*\*:**

**- Implement player mechanics**

**- Develop enemy behaviors**

**- Create basic level structure**

**\*\*Tasks\*\*:**

**1. \*\*Player Mechanics\*\*:**

**- Implement player movement and shooting mechanics.**

**- Develop an armor upgrade system.**

**2. \*\*Enemy Mechanics\*\*:**

**- Define enemy movement and attack patterns.**

**- Implement basic enemy AI.**

**3. \*\*Level Structure\*\*:**

**- Design and implement a basic level with obstacles and pathways.**

**- Develop a system for progressing through levels.**

**\*\*Deliverables\*\*:**

**- Fully functional player movement and shooting mechanics**

**- Basic enemy behaviors and AI**

**- Initial level design with progression system**

**#### \*\*Phase 3: Game Content Development (Days 46-75)\*\***

**\*\*Goals\*\*:**

**- Develop multiple levels**

**- Add sound and music**

**- Refine game mechanics**

**\*\*Tasks\*\*:**

**1. \*\*Level Design\*\*:**

**- Create multiple levels with increasing difficulty.**

**- Design unique layouts and challenges for each level.**

**2. \*\*Audio Implementation\*\*:**

**- Integrate MIDI music and sound effects.**

**- Ensure sound effects are synchronized with game actions.**

**3. \*\*Refinement\*\*:**

**- Improve existing mechanics based on initial testing.**

**- Add visual effects and polish to the game.**

**\*\*Deliverables\*\*:**

**- Multiple playable levels**

**- Integrated audio**

**- Refined and polished game mechanics**

**#### \*\*Phase 4: Testing, Optimization, and Finalization (Days 76-90)\*\***

**\*\*Goals\*\*:**

**- Test and debug the game**

**- Optimize performance**

**- Prepare for initial release**

**\*\*Tasks\*\*:**

**1. \*\*Testing and Debugging\*\*:**

**- Conduct extensive playtesting to identify and fix bugs.**

**- Gather feedback from testers to make final adjustments.**

**2. \*\*Optimization\*\*:**

**- Optimize code for better performance.**

**- Ensure smooth gameplay on target platforms.**

**3. \*\*Preparation for Release\*\*:**

**- Finalize game assets and code.**

**- Package the game for distribution on initial platforms (focus on desktop, plan for mobile).**

**\*\*Deliverables\*\*:**

**- Bug-free and optimized game**

**- Final version ready for initial release**

**### Detailed Weekly Breakdown:**

**#### \*\*Week 1-2\*\* (Days 1-15)**

**- \*\*Day 1-3\*\*: Draft the game design document.**

**- \*\*Day 4-7\*\*: Design player and enemy sprites, environmental elements.**

**- \*\*Day 8-10\*\*: Set up the development environment, install necessary tools.**

**- \*\*Day 11-15\*\*: Create a basic Pygame window and main game loop.**

**#### \*\*Week 3-5\*\* (Days 16-30)**

**- \*\*Day 16-20\*\*: Implement player movement and shooting mechanics.**

**- \*\*Day 21-25\*\*: Develop an armor upgrade system.**

**- \*\*Day 26-30\*\*: Define and implement basic enemy AI.**

**#### \*\*Week 6-7\*\* (Days 31-45)**

**- \*\*Day 31-35\*\*: Design and implement the initial level.**

**- \*\*Day 36-40\*\*: Develop a system for progressing through levels.**

**- \*\*Day 41-45\*\*: Integrate player and enemy mechanics with level design.**

**#### \*\*Week 8-10\*\* (Days 46-60)**

**- \*\*Day 46-50\*\*: Create and implement multiple levels.**

**- \*\*Day 51-55\*\*: Integrate MIDI music and sound effects.**

**- \*\*Day 56-60\*\*: Refine game mechanics and polish visual elements.**

**#### \*\*Week 11-13\*\* (Days 61-75)**

**- \*\*Day 61-65\*\*: Continue creating and refining levels.**

**- \*\*Day 66-70\*\*: Improve and finalize audio integration.**

**- \*\*Day 71-75\*\*: Conduct initial testing and make adjustments.**

**#### \*\*Week 14-15\*\* (Days 76-90)**

**- \*\*Day 76-80\*\*: Conduct extensive playtesting and debugging.**

**- \*\*Day 81-85\*\*: Optimize game performance.**

**- \*\*Day 86-90\*\*: Finalize game assets, package the game for release, and prepare documentation.**

**### Additional Considerations:**

**- \*\*Mobile Version\*\*: Investigate tools for porting to mobile or consider using Kivy for mobile development.**

**- \*\*Feedback Loop\*\*: Regularly test the game and gather feedback to iteratively improve the gameplay experience.**

**- \*\*Community Engagement\*\*: If possible, engage a community of testers early to get diverse feedback and build anticipation for the game.**

**This plan provides a structured approach to developing your game, ensuring that key components are implemented and refined systematically. Feel free to adjust the timeline based on your progress and specific needs.**