William Trojniak, Michael Huan
COS426 Final Project

Pogo Parkour

Abstract:

In this project, we created a 2D platformer where the player traverses a number of levels and avoids spikes in order to reach the end. A core mechanic of our game is the “pogo” mechanic, where the player can jump in midair by pressing an attack key over a spike. Having implemented our own physics and collision detection, we aim to provide a simple yet fun experience for any who chose to play our game.

Introduction:

2D platformers are a very popular subcategory of games where the player controls a character that moves left to right on the screen, usually from a side scrolling perspective. The goal of this project was to design our own 2D platformer and implement the game mechanics from scratch. We designed and created each level ourselves, and we also wrote code to implement the physics of the character’s movement on the screen as well as detect collisions with both solid blocks and spikes. A core mechanic of our game was the “pogo” mechanic. A collision with a spike would normally kill the character and send it back to the start of the level. However, if the player were to press an attack key right above a spike, a sword slash animation would appear and hit the spike, propelling the player upwards in the air. This mechanic gave us many new avenues of creativity for the level design, as spikes represent both danger and an opportunity to progress the level. The project would benefit anyone who enjoys 2D platformers and wants to both relax and challenge themselves with a simple game.

Given the popularity of 2D platformers, there have been a vast number of games built in this style. In particular, the pogo mechanic was inspired by the popular 2D platformer Hollow Knight, where the character is capable of bouncing on top of spikes using a nail slash. Our initial idea of creating a 2D platformer was inspired by the Cave Climber submission in the COS426 Hall of Fame submissions from 2021. There are many different ways to approach creating a 2D platformer. One approach includes keeping the character constant on the screen and moving the background behind it, thus granting the illusion of movement. Another approach is allowing the character to move around on the screen, but allowing the background to move with the character once it reaches a certain bound near the edges of the screen. In terms of frameworks, 2D platformers can be made with a wide variety of frameworks including Three.js, Unity, and much more. Furthermore, though some approaches may be more suited depending on the style and mechanics of the game, the simplicity of 2D platformers allows most approaches to be successful.

For our approach, the first decision to make was what framework to develop the application in. Though many COS426 games used Three.js, which was also a viable option, we decided to go with Pixi.js, a framework optimized for 2D graphics and rendering. For the 2D platforming, we went with the popular method of creating a tilemap for each level. By manually creating a large array of ids, we can create each level by looping over the array and rendering the associated texture with each id. This approach works well in our game because we wanted the level to be strictly constrained to the screen, so the rigid implementation of the tilemap would help us create the level design.

Methodology:

1. Scenes and Levels:

The first piece that we had to figure out was how to implement the scenes and the levels. First of all, we needed to get the assets and art of the game. We used assets from the website opengameart.org, which freely allows game developers to use art for their own projects in adherence to the license terms. We used basic models and animations for the character, the blocks, the spikes, and more. Once we loaded the art into our application, we created an atlas that would store the specific pixels for each block along with an id to identify the blocks. (GET URL OF ART ASSETS) For the level design itself, as described above, we went with a tilemap approach.

  - Goal
    - What did we try to do?
    - Who would benefit?
  - Previous Work
    - What related work have other people done?
    - When do previous approaches fail/succeed?
  - Approach
    - What approach did we try?
    - Under what circumstances do we think it should work well?
    - Why do we think it should work well under those circumstances?
- Methodology
  - What pieces had to be implemented to execute my approach?
  - For each piece...
    - Were there several possible implementations?
    - If there were several possibilities, what were the advantages/disadvantages of each?
    - Which implementation(s) did we do? Why?
    - What did we implement?
    - What didn't we implement? Why not?
- Results
  - How did we measure success?
  - What experiments did we execute?
  - What do my results indicate?
- Discussion
  - Overall, is the approach we took promising?
  - What different approach or variant of this approach is better?
  - Ethical concerns (see above).
  - What follow-up work should be done next?
  - What did we learn by doing this project?
- Conclusion
  - How effectively did we attain our goal?
  - What would the next steps be?
  - What are issues we need to revisit?
-
-

-

References

https://opengameart.org/