1-2022

Panic Engine - A Game Engine

Zachary Winters
Taylor University

PANIC ENGINE

A Game Engine Created by Zachary Winters

1 Introduction

The PANIC engine is a WebGL and three.js based game engine. The base concept and name for this game engine comes from an early iteration with a two-hour deadline, hence the name "PANIC." The project evolved through different iterations, and one is still available to play on the web.

(The original PANIC game)

(Online version of PANIC)

The goal of this project was to implement additional functionality, including input controllers, entity systems/scripting, and collision detection. While the game engine is not fully complete, these are the base structures that need to be in place before additional development.

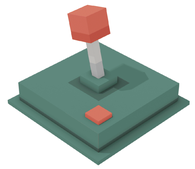
2 Packaging

(Lots of boxes)

The engine's code base is written in JS and structured using ES6 modules. This allows for code cleanliness and easy maintenance, but can cause network issues when importing all of the individual scripts. As a first step, Node and NPM (Node Package Manager) were installed allowing additional packages to be accessed easily.

To fix the network issues, Rollup was installed which builds the project into a single JS file. Rollup's file minification and real-time building also helped development.

3 Input System

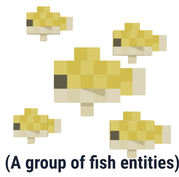


(A joystick gamepad)

When dealing with user inputs, two routes can be taken: either an event-driven architecture or a polling architecture. The old implementation was based on input polling and was extremely limited in its scope. To remedy this, an event management system was created, which handles, emits, and aggregates events.

Using this event system, inputs such as a keyboard, mouse, or game controller can be plugged in and easily detected. This system is accessible by entities within the engine to access and receive data from.

4 Entity System



(A group of fish entities)

Entities within the engine are generated using JSON. These entities were initially limited in their function, but through use of the event system and a new action system, entities can have custom action scripts. These scripts can be bound to input and other events emitted by the engine.

(An example JSON file)

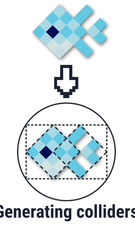
Entities with their new functions required a way to be tracked through the engine. A spatial hash grid data structure was implemented allowing for quick access and simple nearby neighbor retrieval.

5 Collision

(Players colliding!)

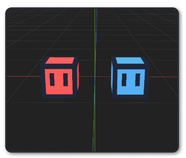
The original collision of the engine was based around the AABB model, but led to problems when rotating entities and introduced clipping. The new system implemented oriented bounding boxes (OBB), allowing for rotated box collisions.

This new system uses the entity JSON format to generate multiple OBBs, a bounding OBB, and a bounding sphere which are checked sequentially to determine collisions.



(Generating colliders)

6 Conclusion



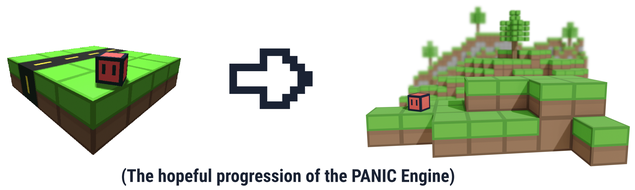
(The new PANIC engine)

The main goal of this project was to get many of the technical aspects finished that I've been dreading. Designing the input system allowed me to explore event management as well as tackling implementing gamepad input. The entity system came together effectively by using the previous event system and linked lists within a spatial hash grid. Collision developments allowed for more exploration into oriented bounding boxes as well as the Separating Axis Theorem for more accurate collision detection. Each of these goals was completed successfully and functionally.

(New version of PANIC)

7 Future Hopes

Revisiting this project has been a joy and brought back many memories of working on past iterations. I hope to continue working on this project as a hobby and implement more new features. I am currently considering a custom scripting language, world creation/loading, and peer-to-peer multiplayer for the future.



(The hopeful progression of the PANIC Engine)

Technologies Used

three.js WebGL npm node

(Three.js, Rollup, WebGL, NPM, and NodeJS)