Alex Agruso, Mauro Alvizo Jr.
Daniel Arreguin, Jake Bram

1 Introduction

Blah blah blah talk about the program

2 Functional Requirements

2.1 User Interface

2.2 Particles and Elements

Physics interactions occur between particles, with each particle being of a certain element. A particle's element determines how it interacts with other particles.

2.2.1 Element Classes

There are four primary classes of elements: solids, liquids, gasses, and powders. Each is described as follows:
Solid: Static and rigid; is not affected by gravity.
Liquid: Dynamic; flows freely but is affected by gravity and collides with other particles.
Gas: Dynamic; diffuses evenly, is not affected by gravity but still collides with other particles.
Powder: Dynamic, awd
We currently plan on implementing 20 different elements

2.2.2 Solid Elements

Wall:
Metal:
Wood:
Glass:
TNT:
Plant:
Rock:

2.2.3 Liquid Elements

Water:
Lava:
Poison:
Oil:

2.2.4 Gas Elements

Smoke:
Fire:
Steam:
Gas:

2.2.5 Powder Elements

Stone:
Sand:
Gunpowder:
Ice:

2.2.6 Miscellaneous Elements

Spark:

3 Non-Functional Requirements

3.1 Event System

Our program will be built on an event system. Any interaction between different components of the program will be handled as an event. There will be many different types of events for our application, many of which will have no common functionality with each other. In order to limit unnecessary abstraction, we will implement the events as an algebraic data type using `std::variant`. We will also use pattern matching with `std::visit`.

3.2 Physics Engine

For stability, the physics engine will be limited to running at 60 ticks per second. Each particle will be represented as an instance of `std::optional` wrapping a particle physics instance. If the particle is empty, or air, it takes the type of `std::nullopt`, while if it has a

4 Sequence Diagram

5 UML Class Diagram