Final Project - (name TBD) - Procedural Graphics

Design Doc

Introduction

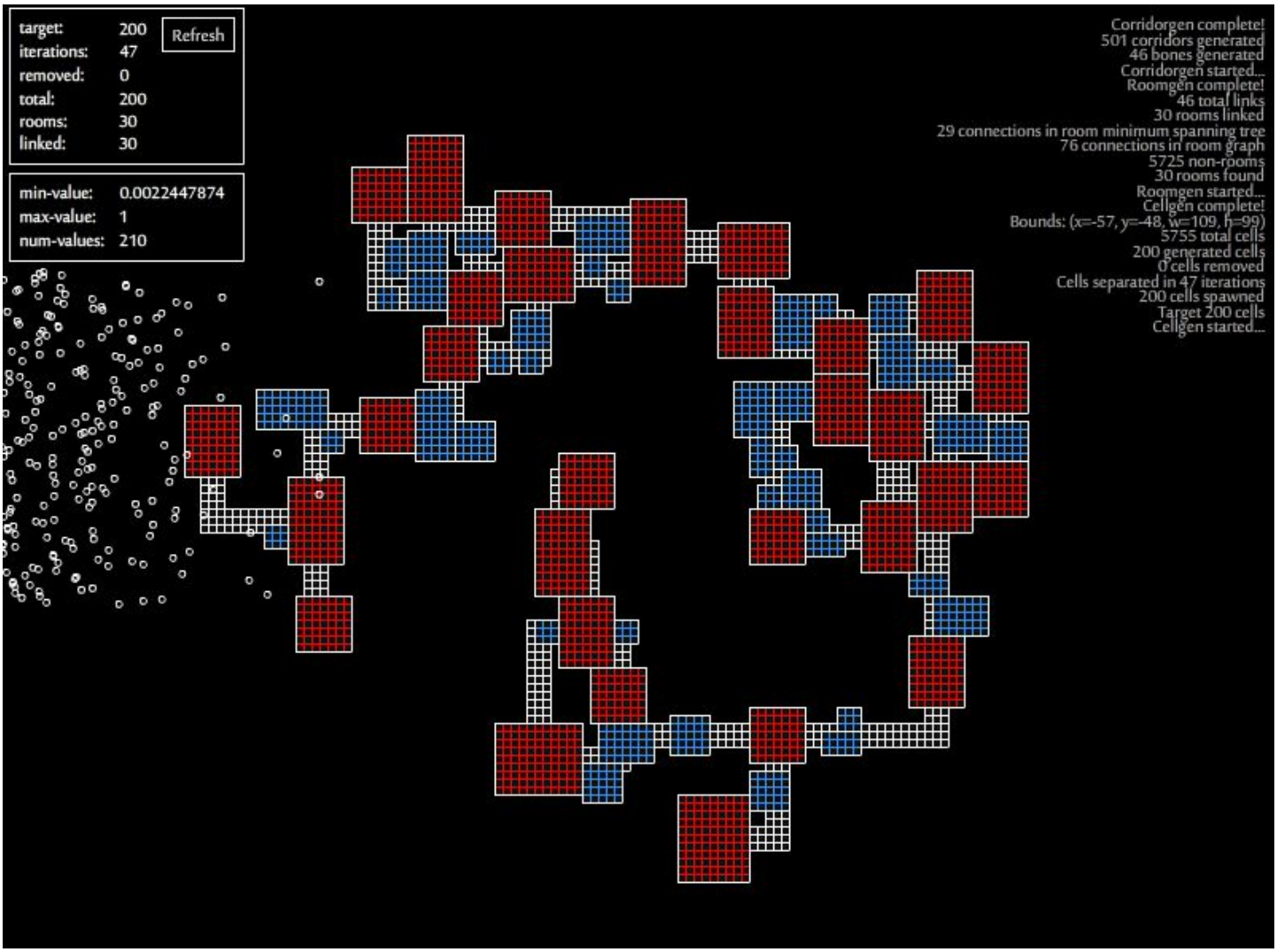
I've been playing games since I was a little kid. Creating interesting levels for games is just a natural direction for my interests to grow into.

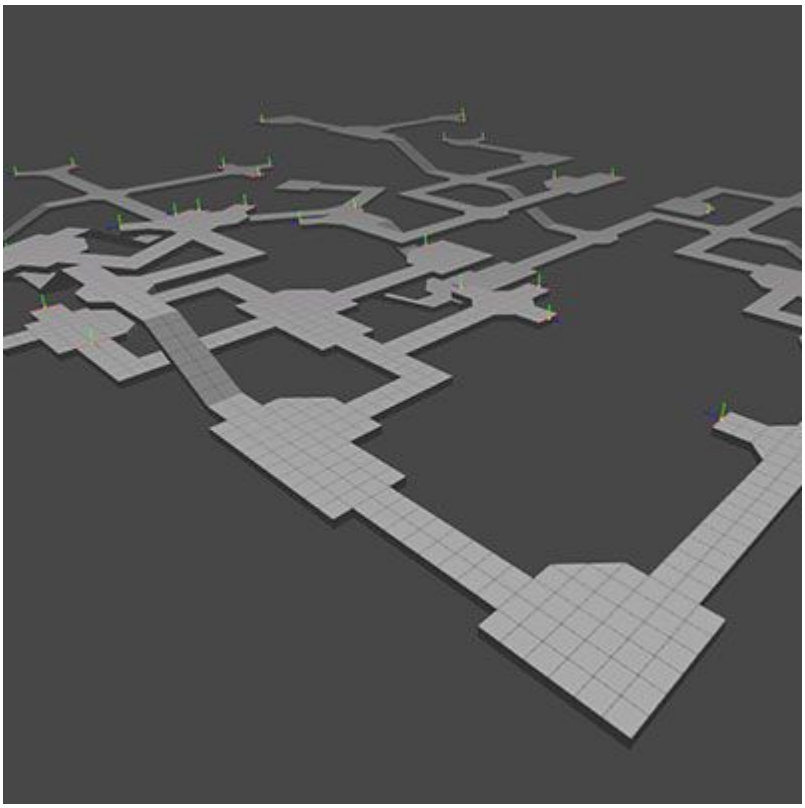
Goal

Create a procedurally generated game level, picture the 2D level maps of diablo but in 3D. Then use voxelized implicit surfaces to actually make the generated levels look more interesting (uneven floors, decorative shapes adorning the rooms, etc). That is the minimum shippable product. If I have more time I will then deal with collision handling for the player (FPS camera).

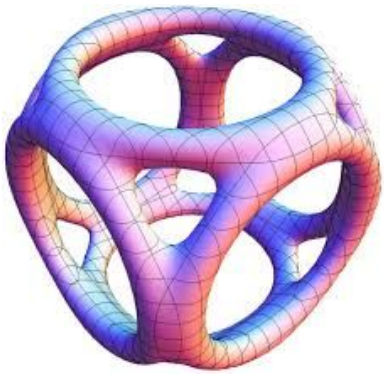
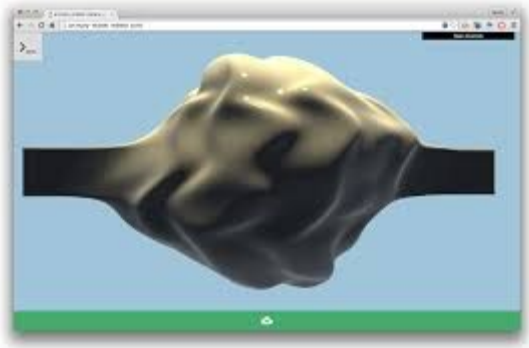
Reference

Level generation





Implicit surfaces

Specification

-> procedural 3D game level (multiple stacks of 2D maps) with a start and end goal.
-> implicit surfaces created via voxels (can get away with it because it's the indie game look)
-> put trampolines in empty spaces so the player doesn't necessarily fall to their doom if they walk off the edge of the map.
The specifications below will be implemented if I have enough time in the final week
-> collision handling so player can't walk through geometry.

Techniques

Level Generation:

-> https://www.reddit.com/r/gamedev/comments/1dlwc4/procedural_dungeon_generation_algorithm_explained/
-> interactive visualization of the above algorithm: http://tinykeep.com/dungen/

The technique above should be good to generate 2D levels. I will then also store the rooms and corridors as instances of a struct that tells me what is connected to what and where the connection is.

To make connections between 2 floors I can pick 2 random rooms in 2 different floors and connect them if they are within some distance to each other. Repeat this process until there are x connections between each floor.

All these rooms can be stored in a grid accelerated structure after they are generated so that later when I create a player, i can simply check for collisions within a radius around the player which would ideally just be with the objects in a single room.

Each floor will have its own grid acceleration for collision detection.

Corridors between 2 floors will be in a separate list. This is because there won't be many connections between floors and knowing the players position should make it easy to identify which floor or between which floors the player is in. Knowing which room the player was in (or closest to) should make it easy to determine which possible corridor or room the player is in to limit collision testing.

Implicit Surfaces:

->http://pcgbook.com/wp-content/uploads/chapter03.pdf -- try

-> https://www.microsoft.com/en-us/research/wp-content/uploads/1982/07/p235-blinn.pdf

-> https://cis700-procedural-graphics.github.io/files/implicit_surfaces_2_21_17.pdf

Create interesting implicit surfaces for each room/corridor using math. Then use Signed Distance Functions to determine the boundary of the implicit surface and generate actual cubes (voxels) just inside the boundary of the SDF of the implicit surface. This is easier than ray marching and the marching cubes algorithm and also makes collision testing easier.

Store the generated cubes in the same grid accelerated structure mentioned above, this way all rooms and corridors have their own grid cell and a list of things in the room with which collisions should be tested with.

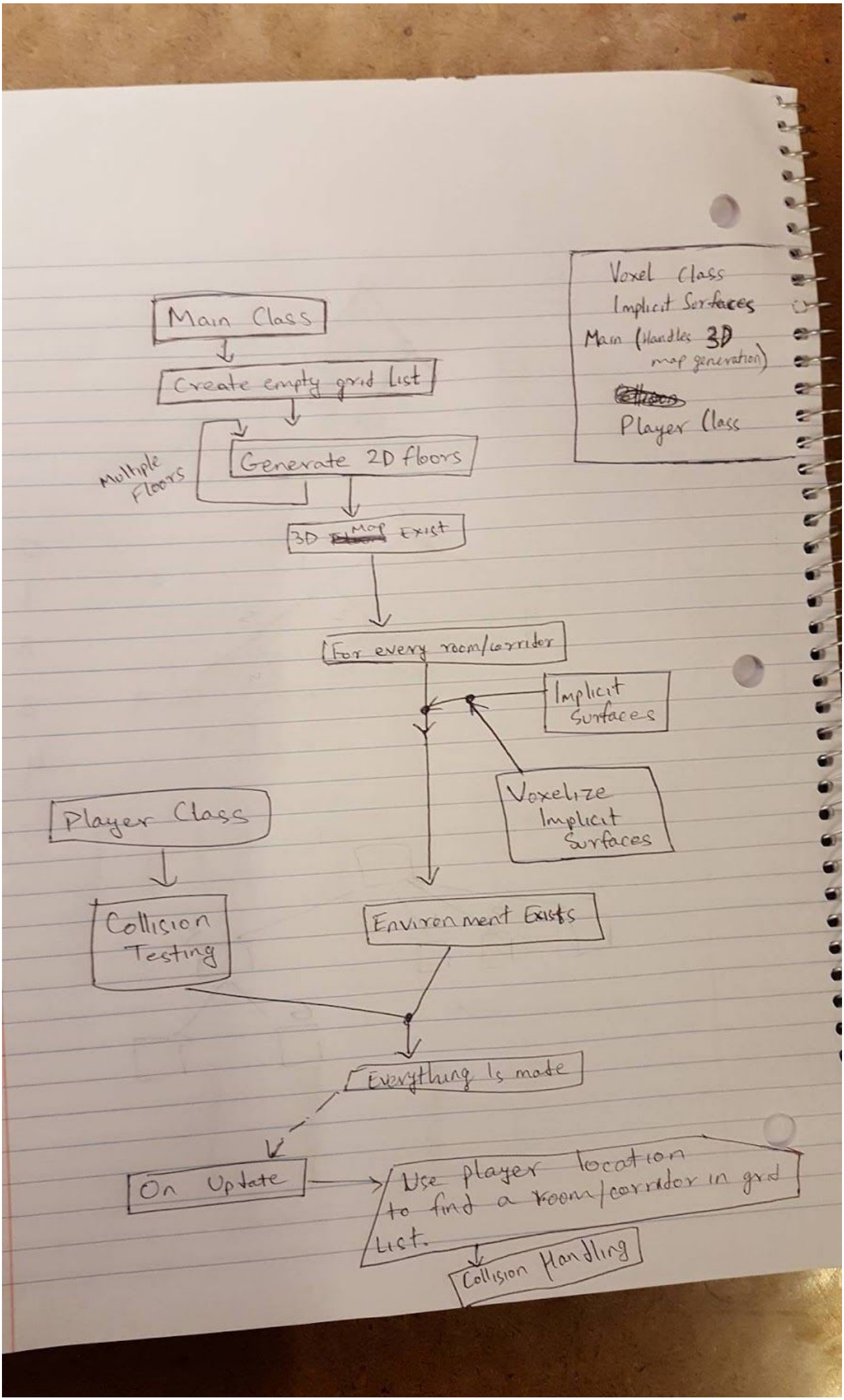
Trampolines:

Load in a trampoline model or use circles to represent them. They just treat the player as a ray, and reflect them. They also increase velocity especially in the y direction.

Collision Testing:

If something is within the sphere centered around the player, move the player in the direction of the negated normal of the player's sphere at the point of collision.

Design


Voxel Class
Implicit Surfaces
Main (Handles 3D map generation)
Player Class
Main Class
Create empty grid list
Multiple Floors
Generate 2D floors
3D Map Exist
For every room/corridor
Implicit Surfaces
Voxelize Implicit Surfaces
Player Class
Collision Testing
Environment Exists
Everything is made
On Update
Use player location to find a room/corridor in grid list.
Collision Handling

TimeLine

Week 1: 2D procedural level generation
Week 1.5: 3D level (stacks of connected 2D levels)
Week 2.5: Voxelized Implicit surfaces
Week 3: Trampolines
Remaining Time: Debugging and possibly collision handling