Application of Voxel-Based Physics Across a Multi-User Network

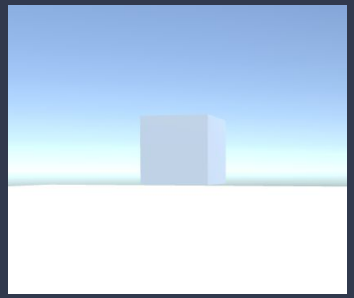
Team Purify
November 21st, 2019
5 P.M. Section
Final Presentation

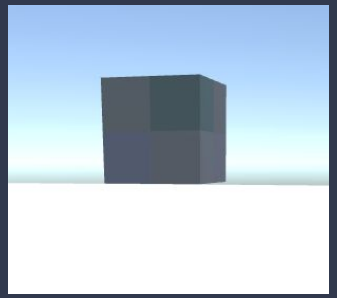
Recap: What is our Project?

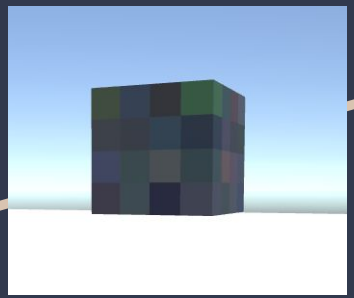
Kinematic Soup

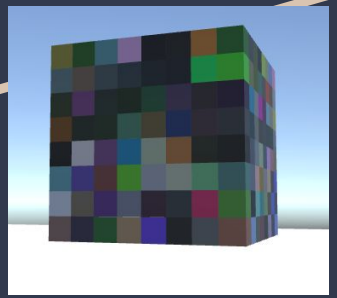
- Creating a destructible environment through the use of voxels that can be communicated across a multi-user network.
- Graphics
  - Octree
  - Meshes
  - Shaders
- Network
  - Multiplayer
- Virtual Reality

Graphics: Octree

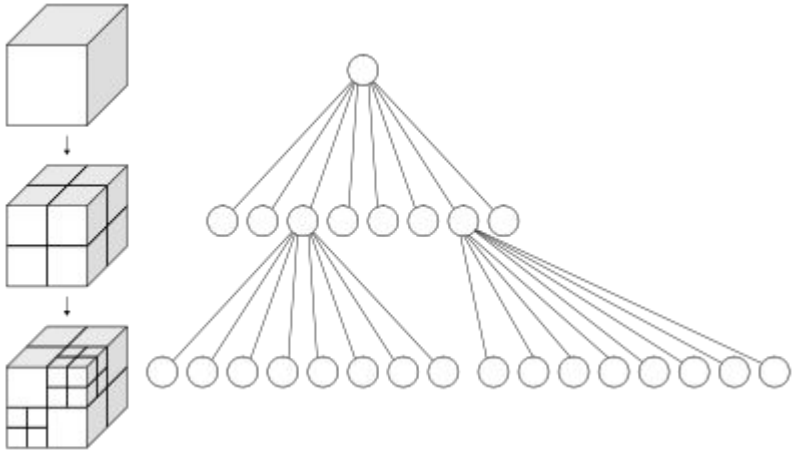




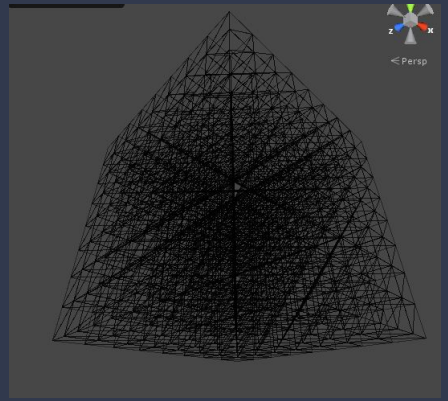


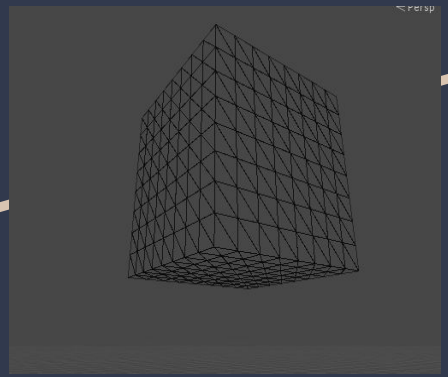


- What is an Octree?
- How did we implement it?
- How/Why does it help us?

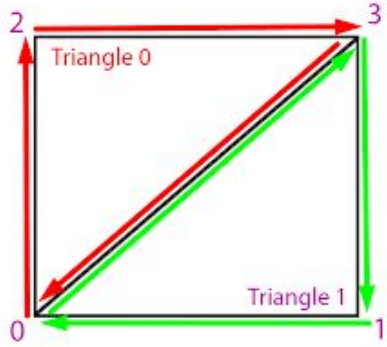


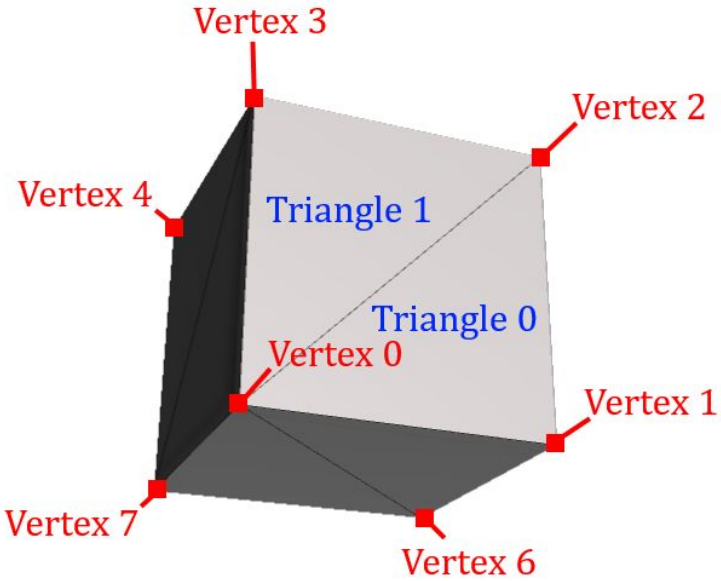
Graphics: Meshing



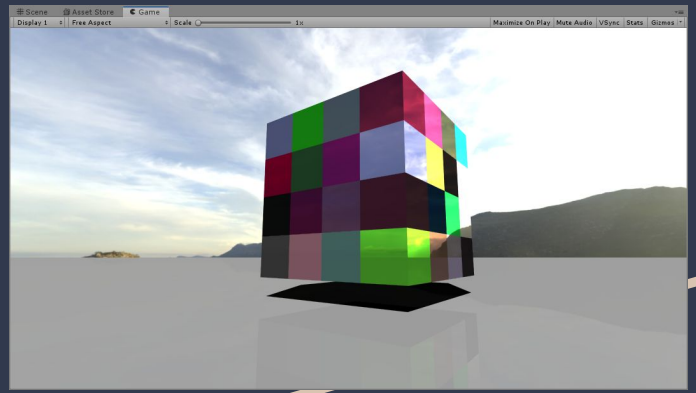


- What is Meshing?
- How does it work?
- Why are we using it?





Graphics: Compute Shaders

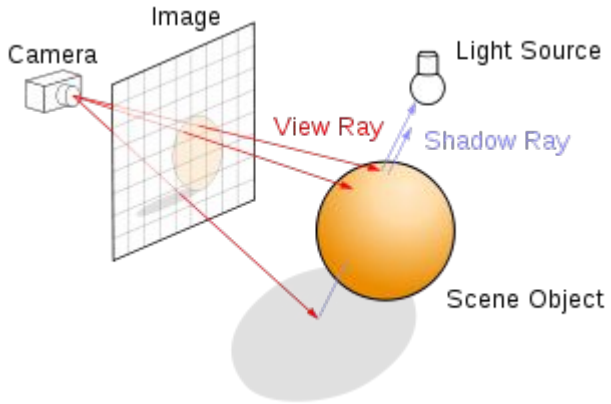


- What are they?
  - Compute shaders utilize a computer's graphics card's computational ability for rendering.
- How do they work?
  - Shaders run on a graphics pipeline that tells the computer how to render each pixel.
  - Often run on multiple threads increasing the speed we can render. (We use 8x8)
- Why are we using them?
  - Create more realistic environments.
  - Improve performance by removing rendering from the CPU.
  - Ray Tracing
    - Reflections
    - Lighting
    - Etc.

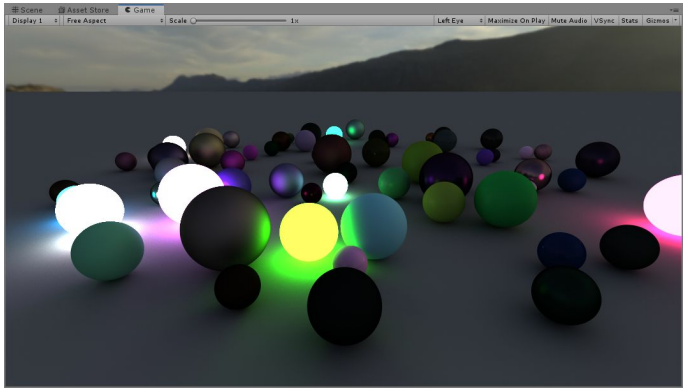
Graphics: Ray Tracing

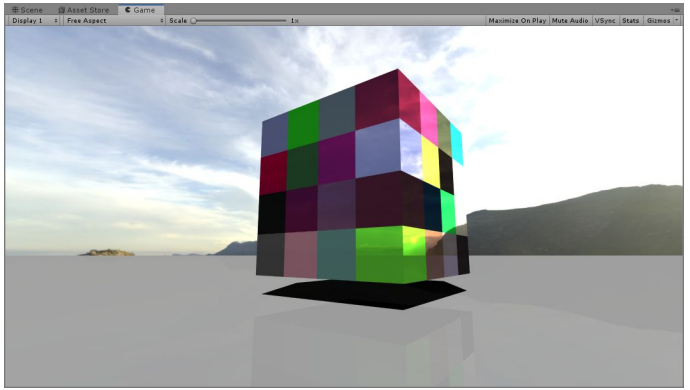
```
void IntersectMeshObject(Ray ray, inout RayHit bestHit, MeshObject meshObject)
{
    uint offset = meshObject.indices_offset;
    uint count = offset + meshObject.indices_count;
    for (uint i = offset; i < count; i += 3)
    {
        float3 v0 = (mul(meshObject.localToWorldMatrix, float4(_Vertices[_Indices[i]], 1))).xyz;
        float3 v1 = (mul(meshObject.localToWorldMatrix, float4(_Vertices[_Indices[i + 1]], 1))).xyz;
        float3 v2 = (mul(meshObject.localToWorldMatrix, float4(_Vertices[_Indices[i + 2]], 1))).xyz;
        float t, u, v;
        if (IntersectTriangle_MT97(ray, v0, v1, v2, t, u, v))
        {
            if (t > 0 && t < bestHit.distance)
            {
                bestHit.distance = t;
                bestHit.position = ray.origin + t * ray.direction;
                bestHit.normal = normalize(cross(v1 - v0, v2 - v0));
                bestHit.albedo = meshObject.albedo;
                bestHit.specular = meshObject.specular;
            }
        }
    }
}
```

- What is Ray Tracing
  - Simulates the steps of light bouncing off surfaces to reproduce realistic lighting and reflections
- How/Why does it help us?
  - Ray tracing is currently being implemented by developers as native support in rendering environments.



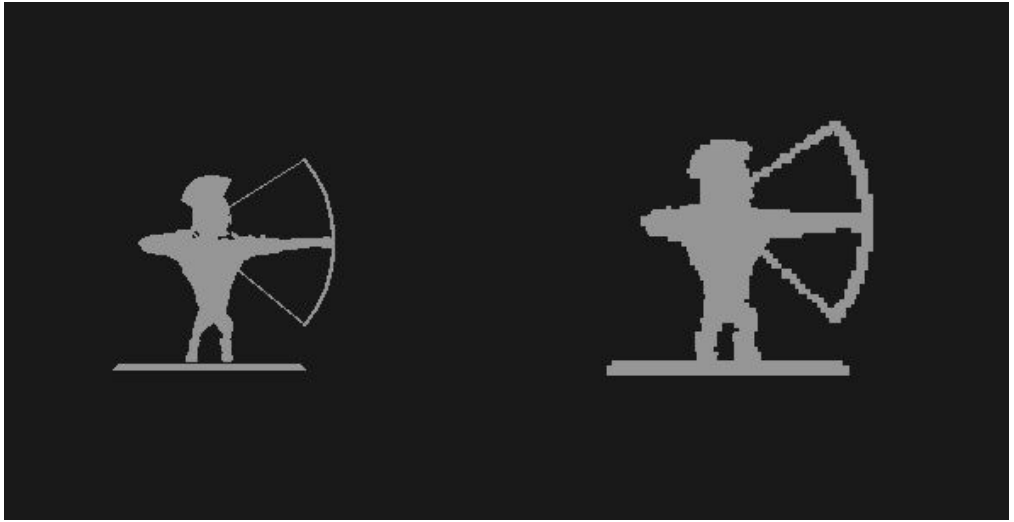
Our Ray Tracing Examples/Tests





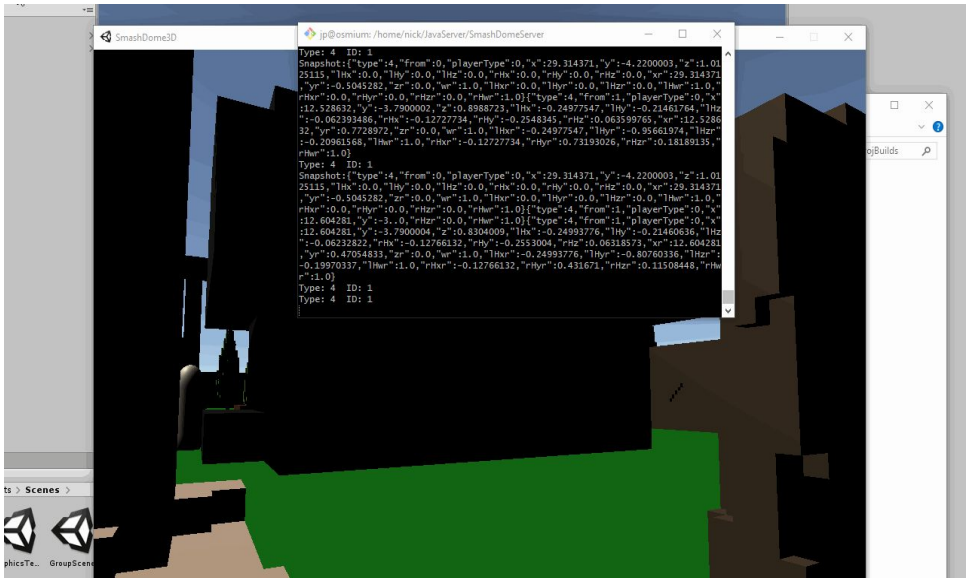
Graphics: Voxelization

- What is it?
- What does it do?
- How can it help us?



Network: Server Implementation

- How the server is implemented
  - Using a Java server for rapid development and compatibility with clients
- What the server is doing
  - Java server makes JSON message and sends it to client where JSON message gets read using C#



Messaging System

Java Server

```
public int type;
public int from; //user
public int playerType;
public float x, y, z;//position
public float lHx, lHy, lHz;//position
public float rHx, rHy, rHz;//position
public float xr, yr, zr, wr;//rotation
public float lHxr, lHyr, lHzr, lHwr;//rotation
public float rHxr, rHyr, rHzr, rHwr;//rotation

public Message(int type)
{
    this.type = type;
}

public Message(int type, int playerType, int from) //request login:privateMsg
{
    this.type = type; this.playerType = playerType; this.from = from;
}
public Message(int type, int from, float x, float y, float z) //send location for player spawn: privateMsg
{
    this.type = type; this.from = from;
    this.x = x; this.y = y; this.z = z;
}
public Message(int type, int from, float xr, float yr, float zr, float wr)
{
    this.type = type; this.from = from;
    this.xr = xr; this.yr = yr; this.zr = zr; this.wr = wr;
}
public Message(int type, int from, float x, float y, float z, float xr, float yr, float zr, float wr,
       float lHx, float lHy, float lHz, float lHxr, float lHyr, float lHzr, float lHwr,
       float rHx, float rHy, float rHz, float rHxr, float rHyr, float rHzr, float rHwr)
{
    this.type = type; this.from = from;
    this.x = x; this.y = y; this.z = z;
    this.xr = xr; this.yr = yr; this.zr = zr; this.wr = wr;
    this.lHx = lHx; this.lHy = lHy; this.lHz = lHz; this.lHxr = lHxr; this.lHyr = lHyr; this.lHzr = lHzr; this.lHwr = lHwr;
    this.rHx = rHx; this.rHy = rHy; this.rHz = rHz; this.rHxr = rHxr; this.rHyr = rHyr; this.rHzr = rHzr; this.rHwr = rHwr;
}
```

C# Clients

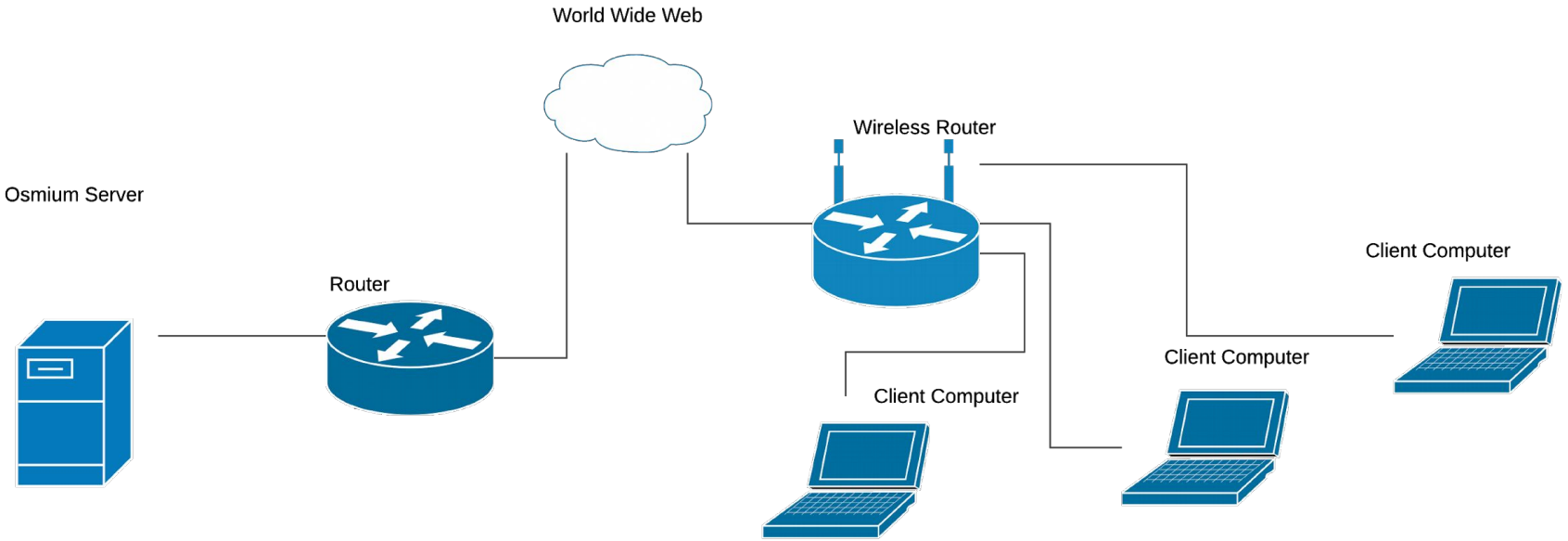
```
namespace SmashDome
{
    [System.Serializable]
    public class Message
    {
        public int type;
        public int from;
        public int playerType;
        public float x, y, z;//position
        public float lHx, lHy, lHz;//position
        public float rHx, rHy, rHz;//position
        public float xr, yr, zr, wr;//rotation
        public float lHxr, lHyr, lHzr, lHwr;//rotation
        public float rHxr, rHyr, rHzr, rHwr;//rotation

        public Message(int type)
        {
            this.type = type;
        }

        public Message(int type, int playerType,int from)
        {
            this.type = type; this.from = from; this.playerType = playerType;
        }

        public Message(int type, int from, float x, float y, float z)
        {
            this.type = type; this.from = from;
            this.x = x; this.y = y; this.z = z;
        }
        public Message(int type, int from, float xr, float yr, float zr, float wr)
        {
            this.type = type; this.from = from;
            this.xr = xr; this.yr = yr; this.zr = zr; this.wr = wr;
        }
        public Message(int type, int from, int playerType, float x, float y, float z, float xr, float yr, float zr, float wr,
            float lHx = 0, float lHy = 0, float lHz = 0, float lHxr = 0, float lHyr = 0, float lHzr = 0, float lHwr = 0,
            float rHx = 0, float rHy = 0, float rHz = 0, float rHxr = 0, float rHyr = 0, float rHzr = 0, float rHwr = 0)
        {
            this.type = type; this.from = from; this.playerType = playerType;
            this.x = x; this.y = y; this.z = z;
            this.xr = xr; this.yr = yr; this.zr = zr; this.wr = wr;
            this.lHx = lHx; this.lHy = lHy; this.lHz = lHz; this.lHxr = lHxr; this.lHyr = lHyr; this.lHzr = lHzr; this.lHwr = lHwr;
            this.rHx = rHx; this.rHy = rHy; this.rHz = rHz; this.rHxr = rHxr; this.rHyr = rHyr; this.rHzr = rHzr; this.rHwr = rHwr;
        }
```

Network Structure



Issues We Faced

Problems:

- Inefficient Octree
  - 4 LOD = 3,072 faces & 4096 Vertices
- Network
  - C++ Network
- Compute Shader
  - Incompatible with Oculus Quest(for now)

Short-Term Solutions:

- Mesh implementation
  - 4 LOD = 384 faces & 576 Vertices
- Changing Languages (until winter break)
  - NodeJS
  - Java
  - Postponing CUDA Processing
- Turn off shader(also for now)

Timeline

Stage 1 - done by 10/31/19
- Create basic graphics environment
- Client/server communication

Stage 2 - done by 11/30/19
- Start basic physics engine
- Allow for multiple-user access

Stage 3 - done by 12/31/19
- Apply Physics to environment manager
- Finalize graphics & physics

Stage 4 - done by 2/29/20
- Optimization of project
- Finalization of all tasks

Stage 5 - Remainder of Spring Semester
- Time accomodation

Demonstration