GPGPU Acceleration of Memworld, a Raycasting Engine for Direct-Memory Representation of 3D Spaces

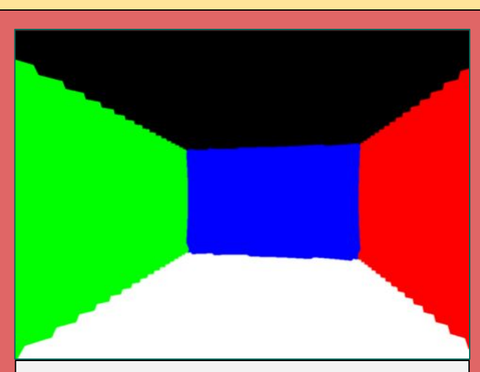
sddec22-18
Client: Dr. Wymore
Adviser: Dr. Wymore

William Blanchard, Mason DeClercq, Jay Edwards, Cristofer Medina Lopez, Dalton Rederick, Collin Reeves

sddec22-18@iastate.edu
https://sddec22-18.sd.ece.iastate.edu

Problem
- Dr. Wymore created a rendering engine called “Memworld”
  - Memworld is a voxel engine (3D pixels/cubes)
  - Used CPU for calculations
  - Slow and “Blocky”

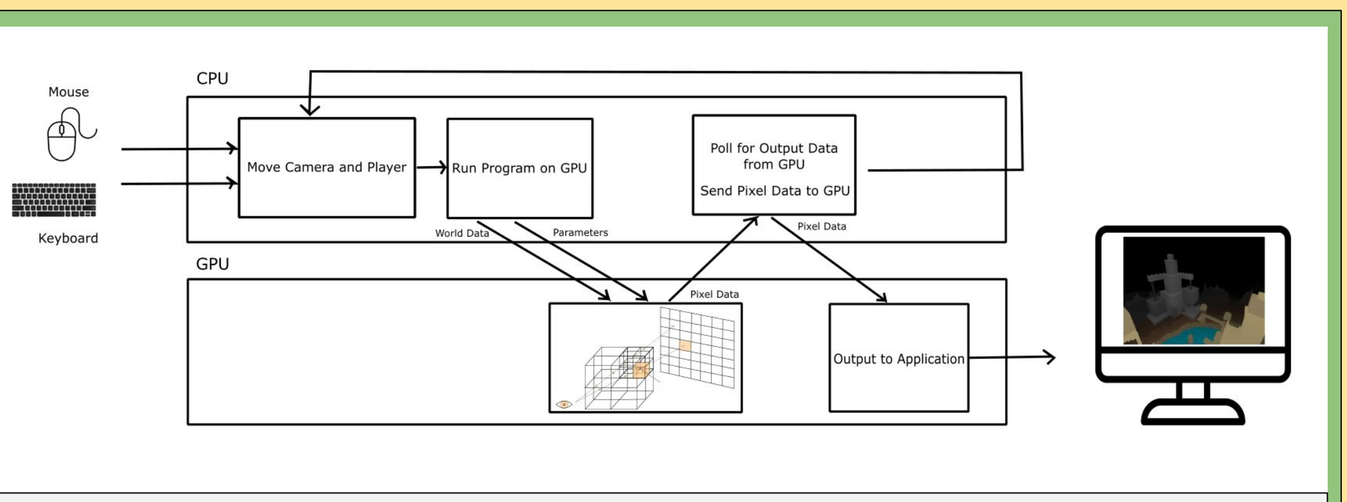
Solution
- GPGPU Parallelization
  - General Purpose Computing Graphical Processing Units
- Include features by his request or to our interest
- Create a simple game to show off improvements and features



The Original Demo

Users
- Game Developers

Use cases
- Render and object/world
- Physics simulation on voxels
- Create a visual representation of current memory storage



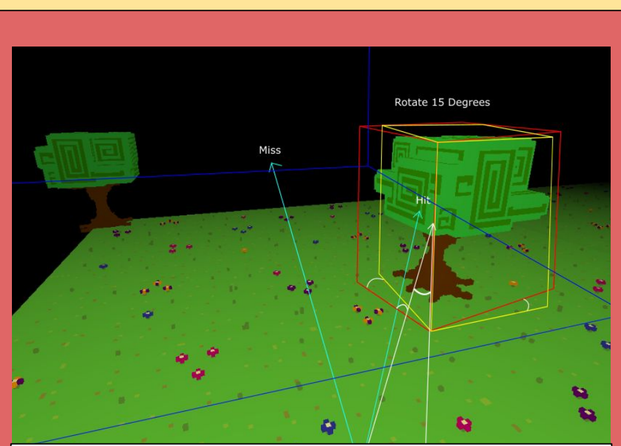
The Conceptual Sketch

Design Approach
Memworld - Main loop, world initialization, shader initialization
File Importer - Called by Memworld to initialize worlds. Imports MagicaVoxel files
OpenCL - Called in main loop, does computation on GPU for displaying the world
Physics - Called in main loop, does computations for collisions, gravity, and moving
UI - Called after OpenCL render has completed, renders text to the screen (star count and fps)

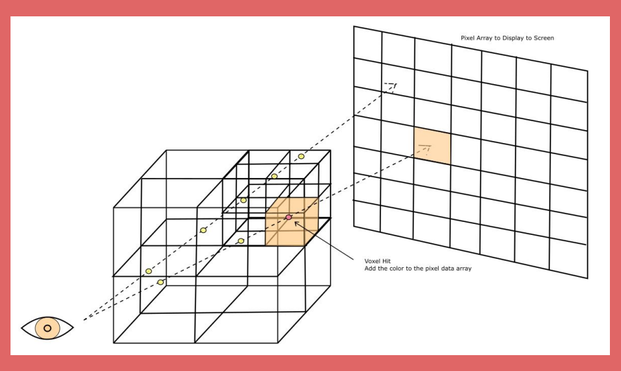
Security Concerns:
File Importing - Overflowing of voxel buffers
Settings - Unintended user inputs

Countermeasures:
File Importing - Only read up until buffer limit
Settings - Ranges of desired user inputs, only read integers

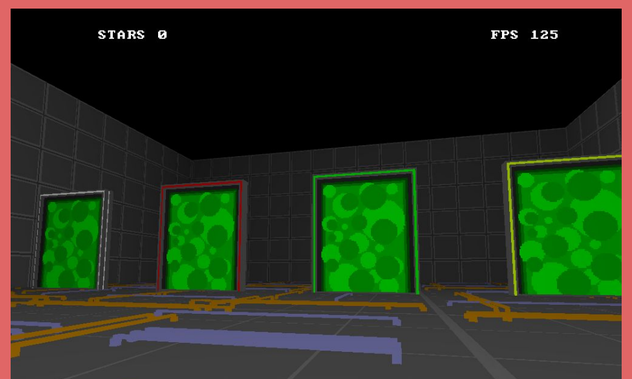
Ray-Based Lighting



Bounding-Box Renderer



3D World to 2D Texture



The Hub World

Average frame rate for worlds:

	NVIDIA GTX 980	NVIDIA GTX 1060	NVIDIA GTX 1070	NVIDIA RTX 3070	Apple M1
World 1	70	67	90	250	213
World 2	100	91	130	275	238
World 3	80	71	100	225	233
World 4	111	143	140	275	226

Requirements/Specifications/Other Constraints
- Must use a direct memory representation/rendering
- Must parallelize rendering algorithm using the GPGPU
- Preferably use a portable GPGPU framework
- Find speedup in terms of FPS
- Minimum 30 FPS with 1024x786 resolution and draw distance of 100 voxels
- Get project working for varied OS and hardware

Test Application
- Must use parallelized engine and should be portable
- Should highlight the advantages and strengths of the engine
- May incorporate new features such as physics, lighting, etc.

Standards (IEEE)
- 730-2014 - Software Quality Assurance Processes
- 1219-1998 - Software Maintenance
- 24748-5-2017 - Life Cycle Management – Part 5: Software Development Planning

Memworld Implementation:
- World Chunks, Moving Objects, and Input Handling

OpenCL Implementation:
- Setup, Main Loop and Teardown
- Adding Objects into World

Renderer Implementation:
- Object Rotation, lighting, and pixel density

Physics Implementation:
- Bounding Box based Collision
- Voxel Based Collision
- Moving Platforms

World Implementation:
- Four Worlds and Hub for 3-D Platformer
- Voxel Dimensions: 512 x 512 x 512
- Collectables and Power-ups
- Designed to Highlight Features of Engine

UI Implementation:
- Display FPS and Collectables count
- 8 x 8 Bitmaps

File Importer Implementation:
- Import Objects using .vox files

Limitations:
- 64 objects, 512 x 512 x 512 voxels, world of 1024 x 1024 x 1024 voxels

Programming Language
- C

Libraries
- GLFW, OpenCL, GLAD

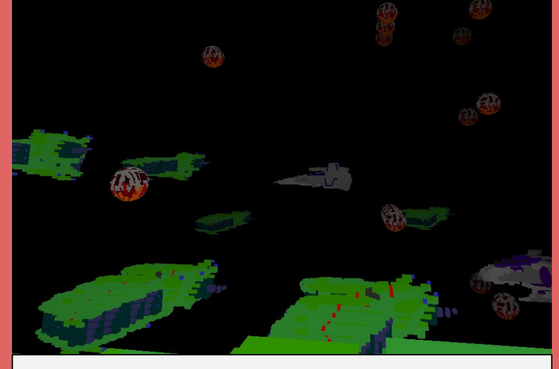
Development Tools/Environments
- Visual Studio Code, Git, CMake, Make, MinGW

Unit and Integration Testing:
- File Importing
- OpenCL
- Physics
- Memworld

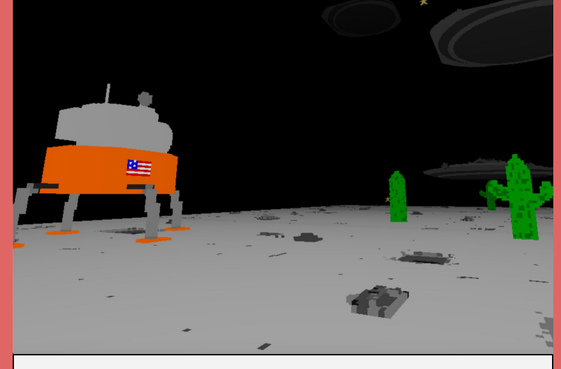
Verified with acceptance testing that all criteria for project was met (frame rate, world size, voxel density)

World 1

World 2



World 3



World 4