**3D Object Scanning and Model Fitting for Unity**

**Team Members**

**Brandon Aulet**

**Christopher Williams**

**Chuck Flowers**

**Mark McCulloh**

**Project Sponsors**

**Peter Smith  
Matt Dombrowski**

# Table of Contents

[Cover Page 1](#_Toc468400511)

[Table of Contents 2](#_Toc468400512)

[Executive Summary 3](#_Toc468400513)

[Overview 3](#_Toc468400514)

[Broader Impact 3](#_Toc468400515)

[Personal Motivations 3](#_Toc468400516)

[Specifications 4](#_Toc468400517)

[Goals 4](#_Toc468400518)

[Requirements 4](#_Toc468400519)

[Research 4](#_Toc468400520)

[Camera Research 4](#_Toc468400521)

[Computer Vision Research 4](#_Toc468400522)

[Unity Game Engine Research 4](#_Toc468400523)

[Detailed Design 4](#_Toc468400524)

[Camera Design 4](#_Toc468400525)

[Computer Vision Design 4](#_Toc468400526)

[Unity Design 4](#_Toc468400527)

[Design Summary 4](#_Toc468400528)

[Camera UML 4](#_Toc468400529)

[Computer Vision UML 4](#_Toc468400530)

[Unity UML 4](#_Toc468400531)

[Overview UML 4](#_Toc468400532)

[Testing Plan 4](#_Toc468400533)

[Camera Testing 4](#_Toc468400534)

[Computer Vision Testing 4](#_Toc468400535)

[Unity Testing 4](#_Toc468400536)

[Budget 4](#_Toc468400537)

[Camera Costs 4](#_Toc468400538)

[Unity Costs 4](#_Toc468400539)

[Milestones 4](#_Toc468400540)

[Summary 4](#_Toc468400541)

# Executive Summary

The purpose of our 3D scanner for Unity is to provide a useful and novel tool for the Unity game engine. This application will be free to use for UCF students and potentially added to the Unity Asset Store for use by indie game developers. The only similar application that is currently available is Intel® RealSense™, which does not offer the 3D scene segmentation features to help developers with level creation that we are planning.

The scanner will consist of three major modules, the first of which accepts RGB-D images from sensors, like the Microsoft Kinect, and preprocesses the data to prepare it for the interpreter. The data interpreter uses this data as input for a computer vision system that will run a scene understanding algorithm to detect objects and estimate object poses in 3D space. The last module will take the information gathered from the computer vision system and transfer this into a format that can be ported into Unity. Then this module will render the appropriate models in a Unity scene.

# Overview

## Broader Impact

What we are creating is more than just a tool to build prototypes of videogame levels. It breaks down many of the barriers of entry to videogame development, softens the learning curve of game design in general, and makes game design accessible to a wider audience. One of the barriers to video game design is the time required to build a game. To build anything of reasonable complexity, a significant investment of time is required to both design the level and then implement it. Our tool aims to consolidate the design and implement stages into a single step. By doing so designs can be quickly evaluated, modified, and revaluated to arrive at the best course of action in as little time as possible. This significantly lowers the barriers of entry to small game studios and single person teams for creating high quality games. This tool will allow them to develop higher quality games without requiring the resources that large game studios have. Our tool also softens the learning curve for learning how to create videogames. The tool eliminates the need to learn any new skill to design levels. This allows individuals who are interested in learning about game design to complete initial projects faster and more quickly evaluate how they feel about the field of videogame design in general. Finally, our tool makes game design more accessible to those who would otherwise not be able to develop videogames via traditional means. By using blocks to design levels rather than writing code or using a two-dimensional drag and drop interface, people with underdeveloped computer skills can engage in videogame design. This means that young children, elderly, and those lacking finer motor skills would be able to play levels of their own creation.

## Personal Motivations

### Brandon Aulet

### Timothy Flowers

I first became interested in working on this project because of the interest I have had in videogames. Although I have never had a very strong desire to work in the video game industry, I have always enjoyed videogames as someone who plays them. When I began my course work as a Computer Science major, I also began to appreciate them on a technical level. When I saw the pitch for this project I thought it would be a great opportunity to exercise both my passions in videogames as well and technical programming.

Apart from my love for videogames, I’ve also been heavily interested in graphics. I think the field presents unique programming opportunities and paradigms that are not often encountered in other areas of computer science. These opportunities include the usage of highly parallelized graphics cards as well as the design of shaders which model the behavior of light when it reflects off different surfaces. I felt like this would be a good project to further explorer my interest in graphics since there would be opportunity to process three dimensional models as well as write an application that interacts with the Unity game engine.

Finally, I found the initial concept of the tool very exciting. When the implementation is complete, it will allow game designers to eliminate the need to spend so much time prototyping levels. Our tool should give them the ability to test an initial concept and then begin to build on top of it. I get a certain satisfaction at building tools for other people to use because I feel that in a certain way, I’m responsible for what people can create with it. To be able to design something that can help others complete a task much more quickly and efficiently than would otherwise be possible is what software engineers should always strive to do.

### Mark McCulloh

### Christopher Williams

I had previously suggested a Senior Design project similar to this, but utilizing procedural generation techniques. I wanted to assist game developers in level creation by creating procedurally generated assets/levels as a tool for the Unity or Unreal Engine. This project gives the same satisfaction in assisting game developers increase efficiency in level prototyping and will allow me to work with the Unity Game Engine as I intended.

I hadn’t considered that my experience with Computer Vision could help with game development and I am excited to apply my experience in this field and learn much more. I am already familiar with resources for potential previous implementations of Computer Vision systems in 3D scene reproduction. I felt that this opportunity could open a door to future game development positions and combines two fields that I am passionate about.

# Specifications

## Goals

## Requirements

# 

# Research

## Camera Research

1.1

## Computer Vision Research

**2.1 – Previous Methods**

We have studied many state-of-the-art computer vision methods for 3D scene processing, object detection, object recognition, and model alignment. Our goal with this research is to find a method or methods to adapt for our application that will provide a fast, accurate, and robust method of processing a 3D scene from our camera and exporting usable information to the Unity Game Engine to create a template level layout for the user.

We ensured our search was broad and included as many different methods as possible to allow for the mitigation of any single method failing or not satisfying the needs of the user. All of the following methods will require significant refinement and alteration to meet our needs but will save us time overall because we will not have to develop a 3D computer vision algorithm from scratch.

Most of these methods provide bounding box information as output after processing. If rotational information is not provided this bounding box gives us the ability to infer where objects are in the scene and allows us to convert this information into a 3D box primitive as our input into the Unity Game Engine. This would work for a physical level built with only rectangular blocks, but we would like to find a method robust enough to include other types of blocks such as cylinders, cones, and pyramids. If a 3D model of the object and sufficient rotational information is provided we can fit other block types within the bounding box. With the appropriate rotations applied this provides a successfully and robustly matched object in the 3D scene space. Other methods match pre-existing 3D models to specific data points in the scene provided.

The limitation set by these model-matching methods would be that users must use these specific types of blocks to get accurate results from our software. This will satisfy our project requirements, but will not make a robust system for broader use. A stretch goal would be to implement more robust methods for alignment that do not rely on pre-existing models. For now, we will adapt one of the model-alignment methods for our software. Any of the methods that require 3D models are appropriate for our purposes because we have been provided 3D models for each of the block types present in our target block set.

**2.1.1 - Learning 6D Object Pose Estimation using 3D Object Coordinates**

**2.1.2 - Aligning 3D Models to RGB-D Images of Cluttered Scenes**

**2.1.3 - Deep Sliding Shapes for Amodal 3D Object Detection in RGB-D Images**

**2.1.4 - Uncertainty-Driven 6D Pose Estimation of Objects and Scenes from a Single RGB Image**

This paper, which debuted at the 2016 Computer Vision and Pattern Recognition (CVPR) Conference, by Brachmann *et al.* is currently our most useful resource for the computer vision interface of our software. The paper is packaged with source code and extensive documentation which allows us to study in-depth what their method is accomplishing and how it functions. This allows us to accurately weigh the benefits and restrictions of this method in comparison to the other methods reviewed.

This algorithm begins by predicting object coordinates and labels with a modified random forest called a joint classification regression forest.

Then Brachmann *et al.* use a stack of forests to generate context information for each pixel in the input image(s).

The object poses are then estimated using Random Sample Consensus (RANSAC). This method is able to perform multi-object detections by obtaining pose estimations for multiple objects and deciding which object the estimations belong to during processing. This is done with the initial predicted values on the input image.

The poses gathered from the use of RANSAC are refined by calculating the distribution of object coordinates in the input image(s). Then the uncertainty levels previously predicted are used to predict camera and object positions when depth data is not available.

**2.2 – Inputs**

There are two basic input formats for the incoming camera data: Point Cloud Data (PCD) or RGB-D image pairs. Point Cloud Data provides millions of data points which provides an implied high accuracy level. The trouble with Point Cloud Data is that minimization or simplification would be required before processing if we wish to achieve fast runtimes.

RGB-D image pairs would contain an RGB image alongside a depth image per frame. This provides a faster runtime more similar to image processing tasks, but it still provides depth information to make sufficiently accurate processing results for our purposes. For these reasons we have chosen to utilize the ability of the Intel RealSense camera to capture RGB-D image pairs for our application.

**2.3 – Datasets**

**2.4 – Outputs**

## Unity Game Engine Research

## 3.1

# Detailed Design

## Camera Design

## Computer Vision Design

## Unity Design

# Design Summary

## Camera UML

## Computer Vision UML

## Unity UML

## Overview UML

## BlockDiagram (2).jpg

# Testing Plan

## Camera Testing

## Computer Vision Testing

## Unity Testing

# Budget

## Camera Costs

## Unity Costs

# Milestones

# Summary