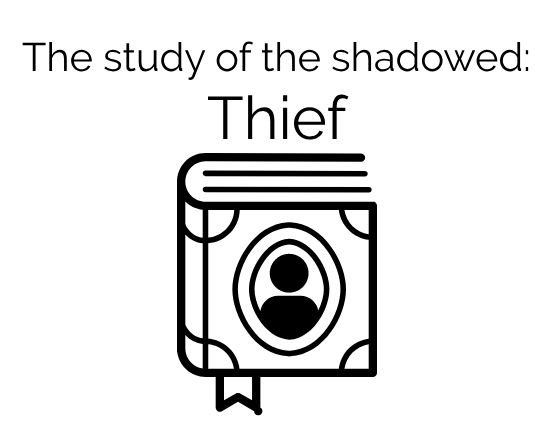
**Senior Project Proposal**



Void Pointer Studios

**Submitted By:**

*Carl Lowther*

carl.lowther@gmail.com

**Submitted To:**

*Calvin Caldwell*

05/20/2018

Version 0.5

# Legal Notice

All information provided by Void Pointer Studios shall be taken “as is” and “as available” with no representations or warranties of any kind, either express or implied, including, but not limited to, any warranties on merchantability, fitness for a particular purpose, and non-infringement. Void Pointer Studios will take all reasonable and available steps in assuring that the contents and services included in this production are accurate and up-to-date. However, it is not in a position to guarantee this at all times. Hence, it shall not be deemed responsible for any potential damage or errors that may occur to the user’s computer because of the latter’s improper or negligent use of this production. No failure or delay by Void Pointer Studios in enforcing any rights or powers it has over its own productions may be deemed a waiver of those or any other rights or powers of the company.

# Copyright Notice

Copyright © 2018 by Void Pointer Studios. All Rights Reserved.

All intellectual property, including but not limited to, text, data, graphics, logos, buttons, icons, images, audio clips, video clips, links, digital downloads, data compilations, and software of this production exclusively belongs to, is controlled by, and is licensed to Void Pointer Studios unless otherwise specified, and is protected by copyright, trademark, and other intellectual property rights provided for by law. No part of this production may be reproduced, distributed, published, transformed, collected, recorded and otherwise exploited without prior written permission of Void Pointer Studios, or as is reasonably permitted under prevailing Intellectual Property Laws. Any breach of these terms shall result in the automatic termination of the use of this production and Void Pointer Studios shall be entitled to all legal reliefs deemed necessary in each instance. All rights, titles, and interests not expressly granted are reserved.

# Revision History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | Date | Author | Company | Comments |
| 0.5 | 5/20/18 | Carl Lowther | Void Pointer Studios | First Draft |
| 1 | 6/6/18 | Carl Lowther | Void Pointer Studios | Logo, H/S platform |

# 

# 

# Signatory Page

Document accepted by:

Calvin Caldwell

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature Date

Document submitted by:

Carl Lowther

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Signature Date

Table of Contents

Table of Contents

[Legal Notice 1](#_Toc516087523)

[Copyright Notice 1](#_Toc516087524)

[Revision History 1](#_Toc516087525)

[Signatory Page 2](#_Toc516087526)

[Introduction 5](#_Toc516087527)

[Purpose 5](#_Toc516087528)

[Scope 5](#_Toc516087529)

[Intended Audience 5](#_Toc516087530)

[Project Management 6](#_Toc516087531)

[Change Management Procedure 6](#_Toc516087532)

[CAT Team 6](#_Toc516087533)

[Medium 6](#_Toc516087534)

[Protocol (response time) 6](#_Toc516087535)

[Impact Analysis 6](#_Toc516087536)

[Archive 6](#_Toc516087537)

[Software Delivery, Installation and Acceptance Criteria 6](#_Toc516087538)

[Documentation and Online Help 7](#_Toc516087539)

[Project Risks 7](#_Toc516087540)

[Customer Responsibilities 7](#_Toc516087541)

[Status Reporting 7](#_Toc516087542)

[System General Description 8](#_Toc516087543)

[Project Summary 8](#_Toc516087544)

[Major Subsystems 11](#_Toc516087545)

[Relation of System to Existing System(s) 11](#_Toc516087546)

[Hardware Platform Description 11](#_Toc516087547)

[Software Platform Description 11](#_Toc516087548)

[Third Party Libraries 11](#_Toc516087549)

[Functional 12](#_Toc516087550)

[**1.** **Player Movement** 12](#_Toc516087551)

[**2.** **Player Interaction** 12](#_Toc516087552)

[**3.** **Enemy Movement, Navigation, and Interaction** 14](#_Toc516087553)

[**4.** **Shadow Detection** 15](#_Toc516087554)

[**5.** **Sound Detection** 15](#_Toc516087555)

[**6.** **Enemy Detection** 15](#_Toc516087556)

[**7.** **Levels** 16](#_Toc516087557)

[**8.** **Database/Server** 16](#_Toc516087558)

[Performance 17](#_Toc516087559)

[Reliability 17](#_Toc516087560)

[Data Description 17](#_Toc516087561)

[Security/Safety 17](#_Toc516087562)

[Constraints 17](#_Toc516087563)

[Glossary 18](#_Toc516087564)

[Appendices 18](#_Toc516087565)

# Introduction

## Purpose

The purpose of this document is to outline and define my project idea for the CSET senior project class at Oregon Institute of Technology. This document will outline the scope, requirements, and design to create 3D Stealth Video Game.

## Scope

The scope of this document is to give a general description of the project and its requirements. This includes the project management, the software that will be used, a project summary and the project requirements.

## Intended Audience

The intended audience for this document includes the senior project proposal instructor, Calvin Caldwell, the developer, Carl Lowther, and anyone who might be interested in the development of this project.

# Project Management

## Change Management Procedure

### CAT Team

Calvin Caldwell and Carl Lowther will make up the CAT Team.

### Medium

All changes should be requested via email to Carl Lowther at carl.lowther@gmail.com and Calvin Caldwell at calvin.caldwell@oit.edu

### Protocol (response time)

Change requests will be analyzed and responded to within three (3) to five (5) business days.

### Impact Analysis

Any change request must go through an impact analysis and review to determine the impact the change will have on the project timeline and the scope of the project.

### Archive

All Change Request Forms will be archived at the end of this document as appendixes to allow for future review.

## Software Delivery, Installation and Acceptance Criteria

This project will be available to play on Windows/OSX in a web browser hosted on a website. The acceptance of this project will be dependent on how closely it relates to the functional requirements of this document.

## Documentation and Online Help

Once the game is started it will show the main menu. The main menu will have a controls page where the user will see all the buttons used and what their use is. The main menu will also have a play game button which will start the game at the tutorial level. The tutorial level is to get the users familiar with the controls and the mechanics of the game.

## Project Risks

There are currently four main identified risks for this project, namely: time restraints, feature creep (getting bogged down with continually adding features), the developer’s limited background in hosting a game on a website, and the developer’s limited knowledge on behavior trees required for realistic enemy AI.

## Customer Responsibilities

N/A

## Status Reporting

Status Reporting will happen weekly to Calvin Caldwell and will include the following:

* Work Completed this Week
* Work to be Completed Next Week
* Issues spotted

# System General Description

## Project Summary

My project idea is a 3D first-person game made in the Unity Engine. The game will be a stealth game, and the main goal is to sneak behind enemies and around levels to obtain gold and valuable items. Sneaking consisting of keeping a distance away from enemies, using shadows in the level to refrain from letting the enemies see the player, and making as little sound as possible. Enemies consist of human guards walking around the level, doing patrols, keeping watch, and if they are alerted, see or hear the player they will try alert other guards and try to pursue and find the player. The game will consist of individual levels, and my initial plan is to have 3 levels including one tutorial level.

Each level will be the size of a bank or small hotel the player should be able to walk across an empty level without enemies within 2 minutes. Each level will have a starting point, where the player begins the level, and an exit point where the player will end the level. Each level may have keys for the player to be able to pick up and unlock locked doors. Each level will also have loot for the player to steal. The objective of each level is to steal gold and valuable objects worth a certain amount of gold (jewels, rings, necklaces, vases, small statues) to reach a certain threshold of gold, which will be around 70-90% of all total value, all while trying to be detected as few times as possible by guards or enemies and to reach the exit point.

To accomplish this, I’ve deemed necessary the 6 main pillars of my project that need to work in order to make it successful, as follows:

1. Shadow Detection
   * Used to detect how far away the player is from a light based on the light’s range and intensity, to see how much the player is in the light (0% completely in shadow, 100% completely visible)
   * Used to determine if enemies are able to see the player
2. Sound Detection
   * Be able to detect how much sound a player is making (0% no sound, 100% running on marble or metal floors) based off of how fast the player is moving and what type of surface they are walking on (e.g. marble floors are louder then grass or carpet)
   * Used to determine if enemies can hear the player
3. Enemy Detection

* First, enemies have “sight” to try and detect the player. First, we start by seeing if the enemy has a clear line of sight to the player and then if we can see the player we use shadow detection to see if the player is “visible” enough to the enemy. Visible being determined by how much the player is in light plus how far away the player is to determine if the player is detected.
* Second, enemies have “hearing” to try and detect the player’s movement and location. Enemies try to “hear” the player by seeing how much sound the player is making plus how far away the player is to determine if the player is detected.
* Third, enemies have proximity detection. Enemies can detect if a player is to close based on how close the player is (in arm's reach) and how fast the player is moving to determine if the player is detected.
* Fourth, enemies have the ability to alert and be alerted by other guards. When the player is detected by an enemy, that enemy will try to alert other nearby guards that they have found, seen, or heard the player. When a guard is alerted by other guards, they will move away from where they are and try to find the player.

1. Enemy Movement, Navigation, and Interaction

* Creating and moving enemies around a level to patrol the level, find the player when they are alerted, or when the player is detected, to chase the player
* Allowing enemies to unlock, open and close doors.
* Turn on and off lights

1. Player Movement

* Creating fluid and responsive actions to move the player:
  + looking around
  + walking and running in 8 directions (North, East, South, West, North East, etc.)
  + crouching down to half the player’s height and to walk slower
  + jumping
  + climbing ladders, ropes, and chest high objects (e.g. walls, fences, windows).

1. Player Interaction

* Player Inventory
* Creating player interactions with each level:
  + opening doors
  + picking locks and safes
  + stealing gold and valuable objects (e.g. jewels, rings, necklaces, vases, small statues) in the level and off of enemies
  + being able to walk up behind non-alerted enemies and when within arms reach press a button to knock out enemies
* being able to put out light sources
* being able to use switches (light switches or switches that can open and close doors, bridges, safes)

My plan for the database is to store a player score for each level. The score is made up of how much gold value that they have stolen, the time it took to complete the level (the timer starting at the beginning of a level and ending when the player has enough gold and has reached the exit point), the times alerted enemies, times knocked out an enemy. For the internet connection, I would like to host the game on a web server so that the game can be played in a web browser, as well as hosting and connecting to the database.

### Major Subsystems

I will be using Unity Engine 2018+ for the development of the game. I will be using Visual Studio for Mac to edit the C# scripts need for the game. I will use a to be determined server hosting, to host the website, the game, and the database. The CSET Git Lab will be used to enforce source control.

### Relation of System to Existing System(s)

N/A

### Hardware Platform Description

The game will be required to run on a computer with minimum specifications of:

Processor: Intel Core 2 Quad Q6600 2.40 GHz

Memory: 4GB RAM

Graphics: GeForce GTX 650 (1024 MB Ram)

### Software Platform Description

The game must run on Google Chrome (Version 66+).

### Third Party Libraries

Assets form the Unity asset store, this will be updated as this project develops.**Product Requirements**

## Functional

### **Player Movement**

* 1. Using the mouse to look around
  2. Using the keyboard to walk, slow walk, and run in 8 directions (North, East, South, West, North East, etc.)
     1. Walking using ‘WASD’ keys
     2. Slow walking using ‘WASD’ and holding ‘left shift’
     3. Running using ‘WASD’ and holding the ‘left control’
  3. Crouching down to half the player’s height and to walk slower
     1. Toggle crouching by hitting the ‘C’ key.
  4. Jumping using the spacebar
  5. climbing ladders, ropes, and chest high objects (walls, fences, windows)
     1. Climbing ladders and ropes by being within arms reach and right-clicking on the object and using ‘WS’ keys to climb up and down
     2. Using the spacebar to jump off the object being climbed
     3. Climb or vault over chest high objects by being within arms reach and right-clicking on the object

### **Player Interaction**

* 1. Player inventory
     1. The player will have an inventory consisting of a list of ordered objects that they can cycle through backwards or forwards one at a time
     2. The player can toggle their inventory by using the ‘G’ key
     3. The player can move forward by one using the ‘T’ key and move backwards using ‘R’
     4. The inventory will consist of:
        1. Lock picks
        2. Keys
        3. A weapon to knock out enemies
        4. The amount of gold collected from that level
        5. Any special items collected from that level
     5. They will be ordered first by the types mentioned at above (2.a.i.1) and then secondly by the order in which they were required.
  2. Opening doors
     1. When in arms’ reach right-click to open or shut doors
     2. If the player is in the way of opening or shutting a door, the door will stop when it reaches the player, requiring the player to click on the door to try again
  3. Picking Locks and safes
     1. The player will have 3 lock picks in their inventory
        1. Each lock will require at least one of the lock picks
        2. Each lock can have up to 4 tumblers. Each tumbler requiring a random pick needed to pick the tumbler
        3. To pick a tumbler the player must be within arms reach and must have a lock pick selected from their inventory and will hold and right-click on the lock. After a short amount of (yet to be determined) time if the player has selected the correct pick the tumbler will make a click sound, else it will make a dull sound.
        4. Once all tumblers are picked the door handle will rotate from its original position signaling to the player that the lock is now unlocked
     2. Safes will have 2-4 dials with each dial having 0-9 possible positions.
        1. The player can find clues through the level for the combination or try to brute force and guess the combination number
        2. To unlock the safe, the player must be in arms’ reach and left or right click on a dial to change the position of the dial, to check if it’s the correct combination the player must right click on the safes handle. If the dials are in the correct position when the player clicks on the handle the safe door will act like a regular door (2.b.) and open
  4. stealing gold, keys, important items, valuable objects (e.g. jewels, rings, necklaces, vases, small statues) in the level and off of enemies
     1. When within arms reach and in line of sight, the player can right click to pick the object up and place it in the player’s inventory
  5. Knocking out enemies
     1. If the enemy is not in an alert state, the player has a weapon selected from their inventor, and the player is behind and within arms reach to the enemy.
        1. The player can left-click and knock out the enemy
  6. Putting out light sources
     1. When in range the player can extinguish torches and fires and can turn off lights by left or right clicking with an empty hand
  7. Using switches (light switches or switches that can open and close doors, bridges, safes)
     1. When in range the player can toggle the switches by left or right clicking with an empty hand

### **Enemy Movement, Navigation, and Interaction**

* 1. Using Unity’s built-in Navigation Mesh builder to create level Navigation Mesh that allows the enemies to know where they are allowed to walk.
  2. Using Unity’s built-in Navigation agent to allow the enemy to traverse and move to a point on the level.
     1. Using a custom behavior tree to tell the enemy where to go and what-Fto-do
        1. The behavior tree will also tell an enemy when to open or close doors and when to interact with things in the world (lights, sitting on a chair or standing up)

### **Shadow Detection**

* 1. How far the player is away from a light based on the light’s range and intensity
  2. If the player is in line of sight to a light, using the distance to a light to see how much the player is in the light (0% completely in shadow, 100% completely visible)

### **Sound Detection**

* 1. Be able to detect how much sound a player is making (0% no sound, 100% running on marble or metal floors) based off of how fast the player is moving and what type of surface they are walking on (Marble and metal floors are louder then grass or carpet)

### **Enemy Detection**

* 1. Enemies have “sight” to try and detect the player.
     1. We start by seeing if the enemy has a clear line of sight to the player
     2. To see if the enemy can see the player we use shadow detection to see if the player is “visible” enough to the enemy.
     3. Visible being determined by how much the player is in light and how far away the player is to determine if the player is detected.
  2. Enemies have “hearing” to try and detect the player’s movement and try to determine the location.
     1. Enemies try to “hear” the player by seeing how much sound the player is making plus how far away the player is to determine if the player is detected
  3. Enemies have proximity detection.
     1. Enemies can detect if a player is to close based on how close the player is (in arm's reach) and how fast the player is moving to determine if the player is detected
  4. Enemies have the ability to alert and be alerted by other guards.
     1. When the player is detected by an enemy, that enemy will try to alert other nearby guards that they have found (seen or heard) the player.
     2. When a guard is alerted by other guards they will move away from where they are and try to find the player

### **Levels**

* 1. Will be the size of a bank or small hotel and the player should be able to walk across an empty level without enemies within 2 minutes
  2. will have a starting point, where the player begins the level, and an exit point where the player will end the level
  3. can have keys for the player to be able to pick up and to unlock locked doors with
  4. will have loot for the player to steal. (Gold, Jewels, Rings, Necklaces, Vases, small statues)
  5. will have guards, with patrols for the guards to follow

### **Database/Server**

* 1. At the end of each level, the player will be able to input their name and it will store their score for that level into the database. No login required, think of old arcade machines
     1. The score is made up of
        1. How much gold value that they have stolen
        2. The time it took to complete the level
           1. The timer starting at the beginning of a level and ending when the player has enough gold and has reached the exit point
        3. The times alerted enemies
        4. Times knocked out an enemy
  2. Using a service like Amazon Web Service to host the Database
  3. Also using a service to host the game on the web using Unity’s WebGL platform, so that the game can be played on a computer without having to download it

### Performance

The game must be able to run at 30 frames per second on the targeted hardware.

### Reliability

The game must run optimally without outside help from the user.

### Data Description

The game should not use over 4GB of ram.

### Security/Safety

All data sent to the server will be optional. If the user chooses to send data, the game will send anonymous scores of how the player did in that level to be put on a public scoreboard for everyone to see. The data may also contain a name if the player wants to put a name to their score.

### Constraints

Cannot be used to make a profit or any revenue by any non-affiliated party.

**User Profiles**

Anyone who uses the game will be classified as a player.

# Glossary

Will be added to as the development will continue. And as requested by the intended audience of this document.

# Appendices