Computer Engineering Program


# OVERVIEW

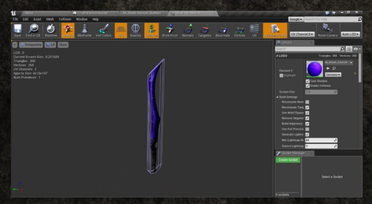
This project is a single-player PC game with realistic 3D graphics and three defining features: Real-time Collision-based Combat, Toggle-able Stealth System, and Reactive A.I. Behaviors.

## Story & Atmosphere

"In the northern, frigid mountains sits an abandoned castle belonging to Thelum, a once powerful but now forgotten arcane kingdom. At its prime, Thelum's expansive territory covered forests, volcanoes, and underground caverns. Legends say that an ancient and powerful arcane artifact still resides within the castle walls. You play as an adventurer, taking on this opportunity for riches, glory and arcane power.

## Combat and Collision

The player, every monster, and every weapon in the game is bounded by a collision mesh. During combat, the player and monsters can attack each other at any time without waiting for turns. Damage is only dealt when the collision meshes of a weapon and a victim overlaps; otherwise the attack misses. These collision meshes move together with character animations, giving the experience of realistic combat.



Editor window showing player sword collision mesh

## Stealth Mode and Invisibility

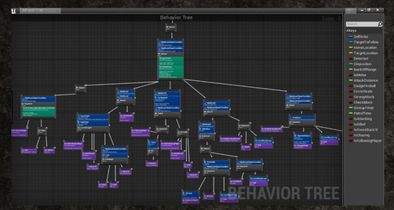
The player can freely toggle in and out of Stealth Mode by pressing Left-Ctrl. While in Stealth Mode, the player moves slower, but sword attacks on unaware enemies deal bonus damage. This Stealth Attack is made easier with the Invisibility Spell.

# MONSTER A.I.

Monster A.I. Behaviors are implemented in a Behavior Tree. Branches, non-leaf nodes, and leaf nodes represent behavior subsets, decisions, and executed behavior tasks respectively.

Only one general Behavior Tree needed, as behavior tasks call Monster Derived Class functions for actual behavior implementation via polymorphism.

State variables in Behavior Tree are stored as Keys in a BlackBoard structure, which are also populated by Monster Derived Classes.




Editor window showing the Behavior Tree

Common Monster Behaviors:

1. ***Player Detection***
   *Detects player within a 90 degree line-of- sight cone in front of monster.
   *Also detects player by sound if player is running nearby.

2. ***Chase and Attack***
   *If detected player, moves to player location until close enough to attack.

3. ***Ranged Backoff***
   *Ranged monsters will retreat backwards if player approaches too closely.

4. ***Investigate***
   *If player ran out-of- sight, monster would go to last-seen location of player and randomly look-and-walk around to hopefully find player again.
   *Monster will also perform Investigate if hit by player's Fireball from long range.

Elite Monster Additional Behaviors:

5. ***Dodge Fireball***
   *Sidesteps incoming Fireball from the player.

6. ***Strong Attack and Circle Strafe***
   *If player is blocking with shield, monster would counter-react with a Strong Attack that is slower to perform but deals more damage.
   *Certain agile monsters such as the Spider will circle around to the player's side or back to bypass the player's shield before attacking.

# GRAPHICS

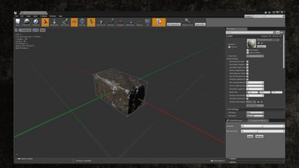
## Snow Material

Materials can be applied to a mesh to control the visual look of the scene. At a high level, they are like "paint" that is applied to an object, but with more rendering capabilities. Using Parent Material and Material Instance, a Material can be parameter-adjusted to appear partially or fully covered in snow.

Adjust parameter to create partial or full snow surface

## Brick Construction

At the end of the Frost Mountain level, contains a "Stone Bridge" terrain object. The walls of this bridge are comprised of individual "Brick" objects placed together compactly. When the level loads, this bridge is constructed via a subroutine which generates and places the bricks.



A single "Brick" object.

The Stone Bridge & walls

The subroutine first calcualte the total number of bricks required, while the placement location of each brick is computed and stored in an array. Then, the brick objects are generated and placed in the game world based on the computed brick placement array. After taking in account the starting location and generation direction for the brick placements, the bridge walls are generated once the subroutine finishes execution.