



## SHANTANU SHRIPAD MANE - GAME PROGRAMMER

Phone No.: +1-385-202-9752 | Email: [shantanu.m934@gmail.com](mailto:shantanu.m934@gmail.com)  
Portfolio: [shantanumane.com](http://shantanumane.com) | [linkedin.com/in/shantanusmane](https://linkedin.com/in/shantanusmane)

### EDUCATION

**University of Utah** - Expected Graduation - May 2019

Pursuing a Masters in Entertainment Arts & Engineering - Game Engineering Track

**K.J. Somaiya College of Engineering, Mumbai, India** - June 2015

Secured a Bachelor of Engineering in Computer Engineering with *First Class Honors*

### SKILLS

**Programming Languages** - C++, C#, Blueprints, Assembly

**Software Experience** - Unreal Engine 4, Unity, Maya, Perforce, Git

**Soft Skills** - Iteration, Collaboration, Problem Solving, Organization

**Computer Science** - 3D Math, Data Structures, Memory & Cache, Code Optimization & Design

**Software Architecture** - UML, Dia

### GAME PROJECTS

**Combat System Project** - Gameplay Animation Programmer - C++, UE4 - Current Project - [Portfolio Page](#)

A combat system similar to that of Bayonetta, focusing on player input and combat mechanics.

- ◆ Created a gameplay and animation system for chain attacks/combos based on input timing and animation events, which is robust enough to allow adding any number of combat moves by designers and chaining between them.
- ◆ Improved responsiveness by accepting next attack input before an attack finishes and later executing the 'Pending Attack'.

**Memory Manager** - Engine Core Programmer - C++ - Oct '17 to Dec '17 - [Portfolio Page](#)

- ◆ Created a memory manager in C++, with Fixed Size & Dynamic Size Allocators, that passes a robust unit test.
- ◆ Implemented Fixed Size Allocators for certain allocation sizes that use arrays of bits to track their memory blocks.
- ◆ Optimized bit operations with Compiler Intrinsic instructions to speed up looking through the bit-arrays.
- ◆ Created a Dynamic Size Heap Allocator to allocate memory of requested size from the reserved heap of memory.

**2D Collision System** - Gameplay Programmer - C++ - Feb '18 to May '18 - [Portfolio Page](#)

- ◆ Created the Collision & gameplay supporting systems for a 2D Game Engine and implemented Pong using it.
- ◆ Implemented the Swept Separating Axis Test for collision checks, and two types of responses to them - block & overlap.
- ◆ Optimized collision system by updating coordinate transformation matrices only for moveable objects, checking collision of only the ball with other objects & responding to only the earliest collision, capitalizing on the game world being sparse.
- ◆ Created libraries of 4x4 Matrix & Vector4 operations for transformations used primarily by collision system.

**Mavrick** - Gameplay Programmer - Blueprints, UE4 - Published May '18 on [Play Store](#) and [itch.io](#)

An Action Game where you pinball and charge at enemies with your fists to send them flying out with an explosion.

- ◆ Setup complete animation state machines for the 'Fighter' and 'Shotgunner' enemies.
- ◆ Worked on the 'Fighter' enemy AI that blocks attacks from the front, needs to be stunned from behind before being able to take damage and can do a short-range charge at the player.
- ◆ Implemented a spawn system allowing to create desired intensity in the game by tuning the difficulty of a set of spawned waves and the kill threshold to spawn every new wave.
- ◆ Designed player abilities and enemies to create intense and high-octane gameplay.

**Warlocks** - Gameplay Programmer - C#, Unity - Aug '18 to Dec '18 - [Portfolio Page](#)

A recreation of a MOBA-esque King-of-the-Hill PvP where you cast spells to fight and defeat other players.

- ◆ Created an input system that can switch between input types - selection & movement, spell-casting & targeting types.
- ◆ Created robust Unit Statistics, Damage and Status Effects systems and pipelines.
- ◆ Implemented a well-rounded spell system with ability interactions, spell-cast types, spell levels, cast times, and cooldowns.
- ◆ Implemented Object Pools to instantiate spells/abilities before game start to eliminate overhead of on-demand creation.