|  |  |
| --- | --- |
| **Shantanu Mane Logo - BW NoName** | **SHANTANU SHRIPAD MANE - GAMEPLAY PROGRAMMER**  **Phone No.:** +91-8591079708 | **Email:** [shantanu.m934@gmail.com](mailto:shantanu.m934@gmail.com)  **Portfolio:** [shantanumane.com](https://shantanumane.com/) | [linkedin.com/in/shantanusmane](https://www.linkedin.com/in/shantanusmane/) |

**EDUCATION**

**University of Utah, Salt Lake City, USA** -*May 2019*

Secured a Master of Entertainment Arts & Engineering - Game Engineering Track with *GPA 3.88/4.00*

**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++, Lua, Blueprints, C#, GLSL, Assembly  **Software -** Visual Studio, Unreal Engine 4, Unity, OpenGL, Maya, MotionBuilder, Perforce, Git, Razor PS4 profiler, PIX profiler  **Soft Skills -** Iteration, Collaboration, Creative Problem Solving | **Game Programming -** 3D Math, Data Structures, Algorithms, Memory & Cache, Code Optimization & Architecture, Blend Trees, Animation Programming, Computer Graphics |

**WORK EXPERIENCE**

**343 Industries - Neal Analytics, Redmond, WA, USA** - *Gameplay Software Engineer - C++, Lua* - Jan ’20 to Sep ‘20

*Halo Infinite - Campaign Engineering team*

* Created a waypoints system to guide players through missions, allowing setup as building blocks and updating with objectives.
* Worked on campaign-related systems, on mission scripts to integrate systemic solutions, improvements to provide design tools and for content completion.
* Implemented a script system to detect player’s engagement in combat to achieve desired mission flow.
* First Responder for Campaign team to investigate, find more information on and route reported bugs.
* Firefighting, workflow support and escalation to ensure productivity and issue resolution for designers & artists.

**SIE Santa Monica Studio, Los Angeles, USA** - *Gameplay Engineer Intern - C++, C#, Lua* - Jul ’19 to Sep ‘19

*God of War: Ragnarok*

* Implemented a root-motion related animation tool feature to make viewing animations in game more convenient.
* Improved a combat collision system to perform more accurate shape intersection tests to better support designers’ vision.
* Optimized a fact-checking system to keep certain types of fact buckets pre-sorted and sort other fact buckets only when necessary which saved 0.2 - 0.3 ms of frame time.
* Fixed bugs related to animation and combat systems.

**Actually A Game Company, Salt Lake City, USA** - *Gameplay Programmer - C++, Blueprints, UE4* - Sep ’18 to May ’19

*Hard Light Vector* - [Released on Steam](https://store.steampowered.com/app/1034740/Hard_Light_Vector/) Mar ’19 - [Portfolio Page](https://shantanumane.com/hard-light-vector/)

An action-adventure FPS game with your fast-paced traversal techniques as tools to conquer giant mechanical monsters.

* Implemented an action-elements system to control on-screen effects and VFX for flair & feedback based on player state.
* Implemented & iterated on the player character’s ‘Thrusters’ that give a small upward boost when you are in-air.
* Worked on an interaction system to indicate and handle interacting with interactable elements near the player.
* Implemented & designed the HUD and various UI elements to achieve a sci-fi feel and power fantasy.
* Contributed to player-side design to create a unique character and resonating abilities that make you feel fast and fierce.

**GAME PROJECTS**

**Combo Attacks System Project** - *Gameplay Programmer - C++, UE4* - [Portfolio Page](https://shantanumane.com/combat-system-project/)

* Created a gameplay and animation system for chain attacks/combos based on input timing, animation events & branches 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’.

**2D Collision System** -*Gameplay & Physics Programmer - C++* - [Portfolio Page](https://shantanumane.com/2d-collision-system/)

* 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.

**Warlocks** - *Gameplay Programmer - C#, Unity* - [Portfolio Page](https://shantanumane.com/warlocks/)

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

* Created a controller system to switch input actions (select, move, target, cast) & handle character states for each action.
* Implemented movement status effects like Stun & Knock-back for spell interactions with characters.
* Implemented a well-rounded spell system with ability interactions & spell target types, levels, cast times & cooldowns.
* Created robust Unit Statistics, Damage and Status Effects systems and pipelines.
* Optimized spells’ Game Object creation by instantiating into Object Pools before game start rather than during gameplay.
* Integrated network functionality for gameplay elements like movement, animation & spells for multiplayer mode.

**Memory Manager** -*Engine Core Tech Programmer - C++* - [Portfolio Page](https://shantanumane.com/memory-manager/)

* 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.