Yiming Pan (He/Him/His)

Gameplay / UI / Networking
Expected graduation: May 2024

Phone: +1 (626)-652-9416
Email: yimingp@smu.edu

Portfolio: https://yimingp.github.io
Linkedin: www.linkedin.com/in/yimingp

Engine/Tools		Programming Languages			Graphics/Audio		Math/Computer Science	Project Dev
Unreal	Unity				DirectX11	FMOD		System Architecture
Perforce	GitHub	C++	C#	HLSL	OpenGL	Aseprite	Linear Algebra	Agile
Jira	Monday	C	Python	SQL	FreeType	Photoshop	Game Math	Scrum
Visual Studio		HTML	Assembly		RenderDoc	Premiere	Programming Paradigm	Debugging

Projects

Asurya's Embers| First Person Bow-And-Arrow Game | Unreal 5.3 2023.08-2023.12
Team of 21 | Roles : UI / BOSS Fight / AI / Player Control / Animation / Pipeline / Audio
Steam Page (Positive)

- [UI] Assisted in building UI components using Unreal UMG system, including main menu, setting menu, pause, tutorial, player HUD, etc. Specifically solving UI input reception bug, maintaining UI focus through different layers, and implementing responsive UI icons for different input schemes.
- [BOSS Fight] Conceptualized and executed the design of a Chinese dragon BOSS battle using Blueprint in Unreal. Involved creating three distinct stages for the BOSS, making various editor triggers and splines for designers to actualized their design. Adding sounds and VFX to result into a realist looking Chinese dragon.
- [AI] Developed the AI logic specifically for the BOSS using Unreal Behavior Tree system. This role also included enhancing the combat dynamics and interface elements for ordinary enemies in the game via Unreal C++.
- [Player Control] Implemented a first-person bow-and-arrow camera control mechanism similar to Skyrim using Blueprint. Created player dash ability similar to DOOM eternal. Utilized multiple Raycasts to enable player actions like vaulting, ledge grabbing.
- [Animation] Contributed to the animation aspects by adding animations for both the player character using Animation Blueprint for building the different hand animation states and the Chinese dragon BOSS using Control Rig for fluid and realistic dragon-snake movements.
- [Pipeline] Was responsible for solving issues with Perforce. Ensured smooth updates of the game engine and used tools like Jira and Monday for effective project management. Also set up nightly builds for consistent build tracking and daily playtesting.
- [Audio] Skillfully designed and integrated a range of environmental and weapon sound effects using Unreal Blueprint, employing Raycast for precise sound variations. This resulted in a highly immersive and adaptive audio experience, tailored to player actions and enhancing the realism during crucial BOSS battle stages.
- [Achievement] Developed and integrated a comprehensive achievement system by researching and utilizing Steam and Epic achievement pipelines, ensuring seamless synchronization of in-game events with players' achievement accounts on these platforms. Also made the art for the achievements.
- [Player Navigation] Developed a navigation beacon system in Unreal Editor to streamline level design and mission guidance, enhancing user interface readaiblity and gameplay fluidity.
- [Dialogue System] Engineered a dialogue system, leveraging Python for dialogue script serialization and enhancing the audio system to streamline dialogue audio management.
- [Input System] Utilized Unreal Enhanced Input System for easy key bindings remap, and identify different states of input keys for different gameplay actions.
- [SDK Ingetration] Implemented and streamlined the integration of Steam and Epic SDKs, establishing efficient build-to-upload pipelines and enhancing client-platform API interactions.

Pans Card | Hearthstone-like Card Game with Scriptable Cards | Custom Engine 2023.08-2024.05

Solo | C++ / UI / System Architecture / Lua Scripting Cards / Networking

- [UI] Built a C++ custom UI framework from the ground up.Supported elements such as Canvas, Buttons, Textboxes, Input Fields, Toggle Switches, Scroll Boxes, and Context Menus. Implemented font rendering using FreeType and SDF (Signed Distance Field).
- [System Architecture] Adopted a composition-based programming approach and the MVC (Model-View-Control) model for decoupling, rapid iteration, scalability, and maintenance; used OOP (Object-Oriented Programming) exclusively for the UI framework. Explore programming paradigm such as Procedural, Functional, and Generic Programming.
- [Lua Scripting] Provided a card editor built with the custom UI framework, allowing players to customize card effects by inputting Lua code. Compiled Lua source code alongside engine code, creating combined C++ and Lua APIs and exposing them for player use, enabling diverse card effects designed by players. As well as enable players to edit local XML file.
- [Networking] Established a server and client version of the game using Winsock2 and TCP protocol, enabling direct IP and port connection for gameplay between two computers. Implemented server-authoritative model and used state synchronization method.
- [Gameplay] Completed basic Hearthstone combat features such as drawing card from deck, play card from hand, drag arrow to attack enemy card or player, and mana system. Enable unique keywords through Lua scripting of the card to do speical gameplay actions like charge, sneak, rush, divine shield, poision, taunt, etc.
- [Animation] Implemented a 2D keyframe animation editor inside the engine for tweening the cards in order to perform draw, attack, die animations.

Custom Engine | Windows 2022.09-2024.08

Solo | C++ / DirectX11 / 2D Physics / FMOD / Multi-threading / DevConsole / Tweening System

- [Overview] Developed games using C++ and Visual Studio on the Windows platform. Utilized WindowsMessage for keyboard input and XInput for controller input. Leveraged the <filesystem> for folder access and I/O operations.
- [Rendering] Supported both DirectX11 and OpenGL rendering APIs, along with 2D and 3D rendering modes. Employed stb_image for texture loading, a custom model loader for OBJ files, FreeType for font files (ttf). Used HLSL for shaders.
- [Physics] Implemented 2D physics including Disc, AABB, Line, OBB, Capsule, Plane, ConvexPoly, ConvexHull, and Raycast detection. In 3D, used Fast Voxel Raycast algorithm for cylinder detection.
- [Multi-threading] Employed a JobSystem template to distribute tasks across different threads and aggregated completed work in the main thread. Enhanced art resource loading efficiency with the engine's built-in multithreading capabilities.
- [Dev Console] Included an configuration-free dev console, facilitating developer debugging via EventSystem-triggered console commands.
- [Tweening System] Integrated a keyframe animation editor (built with the custom UI framework) that allows loading or saving animations locally, easily invoked in game logic for object keyframe animation.

Seafeud | Arcade Racing Game | Unreal 5.1 Steam Page (Mostly Positive)

Team of 48 | Roles: Physics / Performance / UI / Pipeline 2023.01-2023.05

- [Physics] Implemented one-point car physics, using multiple raycasts to simulate physical push and collision.
- [Performance] Worked on optimizing the game's performance; achieved a 15% frame rate improvement.
- [UI] Responsible for the settings menu; contributed to solving Common Input UI focus issues.
- [Audio] Was initially responsible for building the audio asset pipeline. Created interfaces for audio designers to easily swap audio files for different audio trigger to test audio.
- [Pipeline] Was responsible for solving issues with Perforce. Also set up nightly builds for consistent progress tracking and development. Working with leads to set up the pipeline for MUE(Multi-User-Editing).

Pans Tavern | Local Network Multiplayer Restaurant Simulation Game | Custom Engine 2024.01-2024.05

Solo | C++ / Gameplay / Networking / AI

- [Gameplay] Implemented game logic using a 2D top-down camera view, inspired by the rules of PlateUp and Overcook. The game features a tilemap allowing all furniture and ingredients to automatically align to the grid center. Players can freely move and interact with dropped ingredients and dishes, pushing and colliding with them as needed. The game also allows players to choose their character's race and classes, with each race and class offering different attributes and unique gameplay features.
- [Networking] Utilized Winsock2 and UDP for network communication, allowing one player in a local network to host the game, with other players joining by entering the host's IP address and port. Implemented state synchronization and client-side prediction with rollback mechanisms for player movement using Dead Reckoning , ensuring a seamless multiplayer .
- [AI] Developed an AI pathfinding mechanism using Heatmaps and Distance Fields, reducing the need to recalculate paths when no state changes occur, thereby enhancing game performance.
- [Deep Learning] This project really helps me reflect on project planning skill, transition myself into the habit of using project managing tools and methods. Lead me into better understanding of myself, identified what drives me to become a better game developer, and pivoted my mindset in programming.

Pans Survivor Chess | auto-chess-like 2D tower defense game | Custom Engine SteamPage (Downloads: 20,098)

Solo | C++ / Gameplay / UI / Pixel-Art / Rapid-Iteration / Publishing 2023.06-2023.08

- [Gameplay] Implemented key features of auto-chess games including a refreshable and upgradeable shop, a mechanism for leveling up chess pieces by combining three of the same level, and a roster of 42 chess pieces across three tiers and five levels. Also incorporated an enemy spawning algorithm inspired by Vampire Survivors, featuring 30 types of enemies with increasing difficulty based on player survival time and continuous spawning at the edge of the player's vision, thus merging mechanisms from both games into a novel tower defense gameplay.
- [UI] Created basic UI components such as text and buttons for enabling rapid development and integration of user-friendly interfaces that enhanced the overall player experience.
- [Pixel Art] Independently crafted all pixel art assets for the game using Aseprite to craft visually appealing and stylistically coherent graphics.
- [Iteration and Publishing] Rigorously refining the game through player feedback, which significantly enhanced gameplay and user experience. Independently managed the entire Steam publishing process, familiar with Steam SDK and publishing processes.
- [Deep Learning] Publishing a game on Steam presented unique challenges, including managing player expectations and receiving feedback. This experience underscored the importance of resilience and adaptability in game development, particularly when presenting work to a diverse audience.

Education

Southern Methodist University | Dallas Texas

Master's Degree | 2022.08-2024.05 | Master of Interactive Technology - Software Development

[Core Classes] Team Game Production, Math for Games, Game Engine Development.

Bachelor's Degree | 2018.08-2022.07 | Computer Science Major and Creative Computing Minor

[Core classes] Computer Graphics, Machine Learning, Operating Systems, Software Development Project Management, Data Structures, Algorithms, Database, GUI Programming, Programming Languages.

Languages

English (Fluent) Mandarin (Native) Cantoness (Native) Spanish (Beginner)

Hobbies

Video Games, Reading, Writing Fiction, Tennis, Table Tennis, Camping, Cycling, Carpentry