Siniša Komaromi

Senior Game Developer

Location	Croatia
E-mail	sinisa.komaromi@gmail.com
Web	skomaromi.github.io
GitHub	github.com/skomaromi

01 **SKILLS**

Excellent proficiency with the following tools and software stacks:
Unity, **Git**, **MLAPI**, **Photon Bolt**, **GitLab CE**, **Linux**, **Django**

Extensive experience with programming and scripting languages:
C#, **Python**, **C**, **C++**, **PowerShell**, **JavaScript**, **HTML**, **CSS**, **PHP**, **Bash**

Other technologies:
Zenject, **Svelto ECS**, **Unreal Engine**, **Inkscape**, **jQuery**, **Qt**, **PyGame**, **Node.js**, **Laravel**, **D3.js**

02 **EXPERIENCE**

Senior Unity Developer – *Dapper Penguin Studios* **APR 2023 – SEP 2023**

- Implemented new gameplay elements and added features to existing ones, prioritizing minimal performance impact, within a project which utilizes ECS, specifically the Svelto ECS library, in conjunction with Zenject, a DI framework
- Devised and developed a substantial portion of game's visual flow using Unity's UI Toolkit (formerly UI Elements), which involved:
    - implementing code responsible for UI behavior (UX) and layout management;
    - adhering to the reference UI design in Figma, exporting and modifying it as needed to align with game design;
    - fine-tuning USS styling and UXML layout code throughout the process
- Contributed to the development of features which make use of HLSL shader code and the SRP API, simultaneously implementing counterparts in Unity's Shader Graph for comparative analysis when necessitated by issues
- Addressed technical issues and introduced new features using Unity's VFX Graph
- Worked on resolving reported code issues and revised features in response to feedback and evolving design requirements

Senior Unity Developer – *Binx Games* **JUL 2018 – PRESENT**

- Implemented various gameplay features and iterated upon them based on feedback and design updates, for example:
    - a dependency tree which defined prerequisites before a certain building or item was unlocked - implemented both the rule definition code, usable by game designers, as well as gameplay experience and UI code;
    - a mechanism which, based on player’s proximity to a certain in-game region, crossfaded visual effect contribution and defined which types of enemies were spawned
    - a modular system for defining actions upon enemy death

- Worked on gameplay features in an Unreal Engine project, such as:
    - implementing altering gameplay mechanics, adjusting world appearance by manipulating material properties at runtime, and updating the user interface - all in response to an in-game trigger or player input;
    - an enemy inventory system, which allows random weapon selection and weapon unequipping based on changes in world state;
    - creating new and modifying existing enemy behavior trees;
    - integrating visual effects according to design requirements
- Worked on implementing world streaming features, which included:
    - using Unity Addressable Asset System API to load and unload game content;
    - using features of the networking solution to toggle networked objects;
    - splitting content scenes in cells, processing terrain objects to generate low polygon count equivalents, working with third-party assets (MicroSplat) to ensure matching appearance between the stand-in object and the original one as well as using network messaging to ensure both client and server have matching cell scenes loaded
- Established rich CI/CD integration, which included:
    - Unity code which allows defining, customizing and building game presets from the editor;
    - GitLab builder scripting with support for building any of previously defined presets and their deployment to target Steam branch with zero user intervention
- Implemented modular procedural level generation using a simplified two-dimensional random walk algorithm with corresponding editor UI to make preliminary result visualization easier
- Implemented a custom network messaging system for interactable in-game vegetation to reduce network overhead incurred by synchronizing scene vegetation as individual network objects
- Reworked existing world persistence solution to improve performance, usability and store more gameplay parts to the disk
- Implemented a world persistence solution from scratch designed with world streaming in mind with long-running operations (e.g. I/O tasks) spread over multiple frames; later integrated with previously existing game code
- Worked on porting the game from one networking solution to another (MLAPI to Photon Bolt), introducing server authority and client-side prediction for some gameplay features as part of the process
- Adapted third-party character controller solution (Kinematic Character Controller) for a server-authoritative networking library (Photon Bolt) while taking care of previously implemented movement-reliant gameplay code which relied on it
- Implemented support for multiple game store platforms (primarily Steam and Steamworks) to allow for client- and server-side ownership validation

03 **EDUCATION**

Master of Computer Engineering – *University of Osijek, Croatia* **OCT 2017 – SEP 2019**

Developed various applications and games as part of course assignments, some of them leveraging network for its functionality, including:

- a 2D side-scrolling game with procedural level generation and voice control (Unity, C#)
- a simple turn-based rogue-like game (Python, PyGame)
- a real-time chat application with client and server counterparts with decentralized media storage (Python, WebSocket, Java, Django, REST, IPFS)
- an application which processes and displays satellite imagery in the form of a web map (Python, GDAL, Rasterio, Django)
- a single-page responsive web application for posting classifieds (Python, Django, REST)
- an image viewer extensible with user-created filters, was used as a part of a university image processing course (Python, Qt, OpenCV)
- an interactive map of Croatia displaying changes of indicators over time (HTML, CSS, D3.js)