Software Development Process (SDP)

Team: Engineering Simulations 4
Contributors: Cameron Paul Crutcher, Robert Orlin Jacobson, Levi Z James, Alexander Yang Rhoads
Class: CS46X

Principles

- Consistent asynchronous communication, with answers expected within 24 hours using Discord and Microsoft Teams.
- We will use GitHub Issues to create tasks for the team to track. Issues must be assigned at creation.
- We will use GitHub Projects with a Kanban board to continuously work on the backlog.
- New development branches shall be created for individual tasks. Clean branches after successful merging on a weekly basis.
- Pull requests are created whenever work is done on a feature, bug, or refactoring.
- Pull requests have to be reviewed by at least one member before they are merged.
- Pull requests must follow the Definition of Done and be linked to a work item.
- Work items should take at most two days to complete and need to encompass a significant portion of work. If an item is too large to complete in this time, then it must be refactored into smaller items.
- Unexpected challenges shall be posted as soon as possible within team communication channels. Each challenge shall be assessed as a collective effort and timelines shall be updated accordingly. All stakeholders will be notified.
- The software development process may be updated provided all necessary parties have accepted the changes. Ensure this document is adjusted accordingly.

Process

Our software development process follows a hybrid Agile and feature-driven development approach, with emphasis on constant iteration and interaction with the project partner. The feature-driven approach is embodied by the use of a Kanban board for tracking feature development. By the nature of the project requirements, the team will be accommodating DevOps development features, given the small team scale, which necessitates multi-discipline roles.

The process includes weekly meetings with stakeholders and Kanban categorization for features in various development stages. Team members will operate asynchronously of each other and use designated discussion channels for additional questions, concerns, and input. Backlog management will be completed post-stakeholder meetings (Thursday), with check-ins at the beginning of the week (Monday). The team will have 48 hours to respond to pull requests and issues.

Roles

Team leads, who provide management for stakeholder meetings and progress reports, will rotate on a weekly basis. All members, including the week's assigned team lead, are expected to contribute to software development and provide input to the stakeholders for assessment of the current project status on an individual basis. The team as a whole will handle project management.

Software development roles included are as follows:

- Environment artist(s): Responsible for terrain generation through photogrammetry pipeline from images to rendered 3D environment using Meshroom and Blender.
- Simulations engineer(s): Responsible for ensuring the rendered environment interacts with the vehicle simulation accurately. Uses Bevy game engine and associated libraries to accomplish this functionality.
- Vehicle simulation designer: Responsible for ensuring pre-existing car simulation used with the project is optimized as needed to perform to our software and hardware specifications.
- UI/UX Engineer(s): Responsible for managing user interface with an emphasis on ease of access and usability without compromising functionality.
- Software Tester(s): Responsible for testing the simulation software prior to demo. Requires access to suitable hardware capable of running the simulations.
- Web Developer(s)(Optional): Responsible for deployment of the application software online as a stretch goal. Includes front and back end development.

Tooling

Version Control	GitHub
Project Management	GitHub Issues and Projects.
Documentation	https://github.com/withastro/starlight
Test Framework	Playwright (for Web Deployment stretch goal), Cargo — Rust package manager
Linting and Formatting	Prettier
CI/CD	GitHub Actions
IDE	Visual Studio Code
Graphic Design	Figma
Others	AI assistants, creation of code executable

Definition of Done (DoD)

- Meets the acceptance criteria.
- Functional requirements are met
- Non-functional requirements are met
- Changes and code have been peer-reviewed
- All tests are successful.
- Changes get merged into the main branch.
- If they fit, changes are to be carried over to all components
- No regressions from the current iteration of the main branch.
- Development branches scrubbed after successful merging
- Relevant documentation updated, including deployment instructions and release notes.
- Breaking changes are to either be avoided or evaluated.
- Any changes are carried over to staging.
- Able to be applied to a demo for the next stakeholder meeting.
- Complete software testing has been performed on a minimum spec system

Release Cycle.

- Ensure local development branches are deployed to remote repo at a minimum of once a day (after development has occurred)
- Automatically deploy to staging every merge to main branch
- Release every three months for first two releases
- Final release is six weeks from previous release
- Use semantic versioning `MAJOR.minor.patch`
    - Increment the `minor` version for new features
    - Increment the `patch` version for bug fixes
    - Increment the `major` version for breaking API changes
    - Major will be 0 until the API is stable.

Environments

Environment	Infrastructure	Deployment	What is it for?	Monitoring
Production	Google Cloud through CI/CD (if stretch goal of web deployment is met) Local (stretch goal not met)	Release	Exploring the car simulation on personal rigs	Sentry
Staging (Test)	Local (macOS, Windows, Linux)	Pull Request	New unreleased features and integration tests	N/A
Dev	Local (macOS, Windows, Linux)	Commit	Development and unit tests	N/A