Contribution Guide

If you're interested in submitting code to this gamemode, read this guide carefully.

Required Tools

- **Visual Studio Code**
  - Text Editor for developing OverPy source code
- **OverPy**
  - VScode extension for compiling OverPy source code -> workshop script
- **Git**
  - Version control system
- **GitHub**
  - Remote archive of this source code repository
  - Issue tracking and Pull Request
- **Discord**
  - Communication between developers

Setup Instructions

1. Download Visual Studio Code.
2. Go to extension tab in VScode and install OverPy extension.
3. Clone the repository to your local computer (use GitHub Desktop if you don't know how to clone using terminal).
4. Open cloned repository folder on VS Code.
5. Try building gamemode using instructions below.
6. Follow general workflow to start development.

Building

For instructions on how to use OverPy, see here

1. Compile `main.opy` using the compiling instructions at OverPy Wiki.
2. Open a custom game in Overwatch.

3. Paste the compiled gamemode code.

General workflow

1. Find an issue to fix (or submit one yourself).
2. Create new branch dedicated to that issue.
3. Modify codebase using VScode. Be sure to follow styling guidelines.
4. Compile and test new codebase.
5. Commit working changes.
6. Submit Pull Request (PR) to merge your branch into `dev` branch.
7. Wait for owner to approve and merge your Pull Request.

Release Cycle

- For every 3~5 features added to staging branch, the devs will compile a staging build and test each feature manually.
- If your feature fails to meet quality expectations, the MR corresponding to that feature will be reverted to drop the changes.
- After each feature in staging branch is confirmed, the staging branch will be merged to main branch and released as latest build. If the latest build plays without crashing, it will be set as the stable build.

Coding Style Guidelines

This codebase follows Python's PEP 8 style guide since OverPy follows Python syntax.

- Variable names should be snake_case. Example: `hero_health_armor`
- Function names should be camelCase. Example: `destroyBarrier()`
- File names should be snake_case. Example: `custom_heroes.opy`
- No magic numbers. Constants should always be referred to by the names defined in `src/constants`.
- Rule and subroutine names should follow the format rule `"[file_name.opy]: My custom rule"` and `@Name "[file_name.opy]: mySubroutineFunc()"` respectively.
- Keep rules simple; each rule should only perform one task. Try to limit rules to at most 10 lines of code and group large blocks of code into subroutines whenever possible. See `src/heroes/bastion` as an example.

Revision #1
Created 21 June 2024 16:33:45 by Admin
Updated 20 August 2024 22:38:14 by Admin