

# ISOMAKER

## README



Epitech EIP 2024/2025  
Tom Bariteau-Peter, Léa Guillemard Alessandro Tosi

IsoMaker is a game engine for creating isometric games. It is written in C++ and uses the Raylib for rendering.

## C++ CODING STYLE

The foundation of the coding style applied to this project is a C++ adaptation of Epitech's C coding style, here are the main rules:

### NAMING CONVENTIONS

- All subdirectories and files under the /src and /includes directories should have one word names when possible. These, as well as classes, should follow the UpperCamelCase naming convention, which dictates every compound word in a name is capitalized.
- Interfaces and abstract classes must have names starting with a capital I and A respectively (e.g., IHandler, AHandler).
- Namespaces, functions and variables should follow the lowerCamelCase naming convention, which dictates the first compound word should start with a lowercase, and every compound word after that should be capitalized (e.g., input::KeyboardHandler::startLoop(bool status)).

### PROJECT FOLDER ARCHITECTURE

- .hpp files containing interfaces, abstract class implementations, project specific variable types and other such things should be found in /includes/<subdirectory> (e.g., IHandler.hpp, AHandler.hpp and Types.hpp in /includes/Input).
- .cpp source files required for building the project should be found in /src/<subdirectory>, along with their respective .hpp files. (e.g., Keyboard.cpp and Keyboard.hpp in /src/Input).

### FILES

- All files should start with the standard Epitech header provided by the official Epitech header extension on VSCode.
- Contents of .hpp files should be preceded by #pragma once.
- All files should end with \n (newline).

### CLASSES

- A class' access specifiers should be written in the following order: public, protected, private.
- Function prototypes should be declared before variables.

# COMMIT MESSAGE STANDARD

All commits contributing to this project must be accompanied by a commit message that adheres to the following format, based on the Conventional Commits standard:

<type>(<scope>): <description>

## STRUCTURE

- **type:** Describes the purpose of the commit. Use one of the following predefined types:
  - **feat:** A new feature.
  - **fix:** A bug fix.
  - **refacto:** Code changes that neither fix a bug nor add a feature.
  - **doc:** Documentation changes (e.g., README updates).
  - **style:** Code style changes (e.g., formatting, missing semicolons) that do not affect functionality.
  - **test:** Adding or updating tests.
  - **build:** Maintenance tasks (e.g., build scripts, dependency updates).
- **scope:** Identifies the specific area of the codebase affected. Keep it concise, such as:
  - A module name (e.g., graphics, input handling).
  - A component name (e.g., button, keyboard handler).
  - Use \* for a global change affecting multiple areas.
- **description:** A brief, imperative summary of the change.
  - Start with a verb in the present tense (e.g., add, fix, update).
  - Don't capitalize the first letter or end with a period.

## Examples

**Adding a new feature:** feat(input handling): add support for joypad controls

**Fixing a bug:** fix(graphics): resolve crash on object rendering

**Updating documentation:** doc(readme): improve commit convention examples

**Refactoring code:** refacto(engine): simplify event loop logic

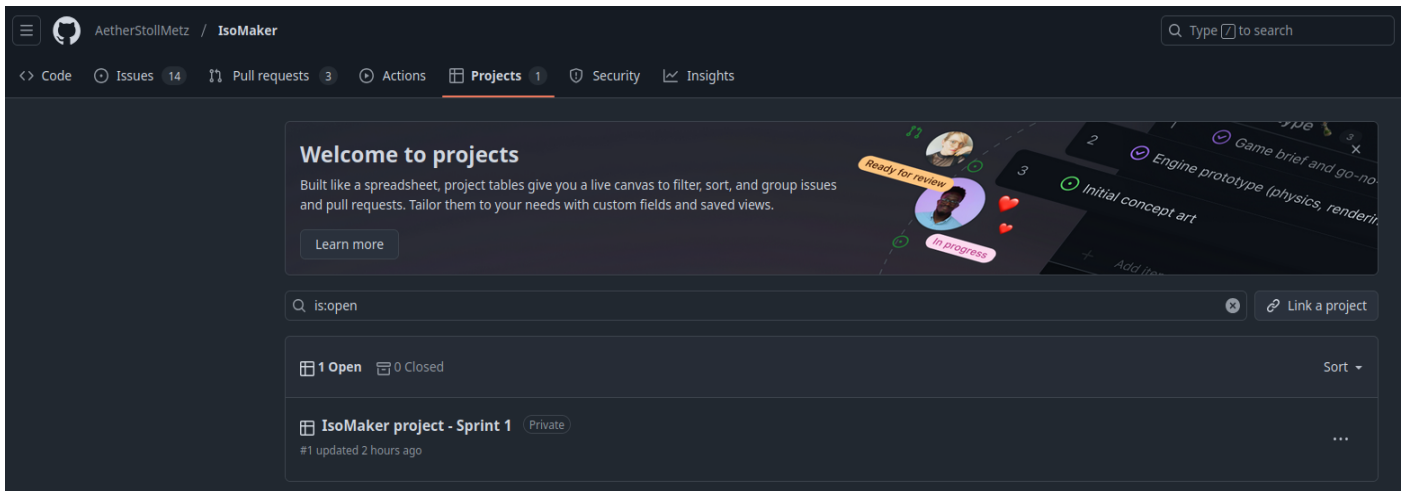
**Updating tests:** test(graphics): add tests for sprite rendering

## WHY USE THIS FORMAT ?

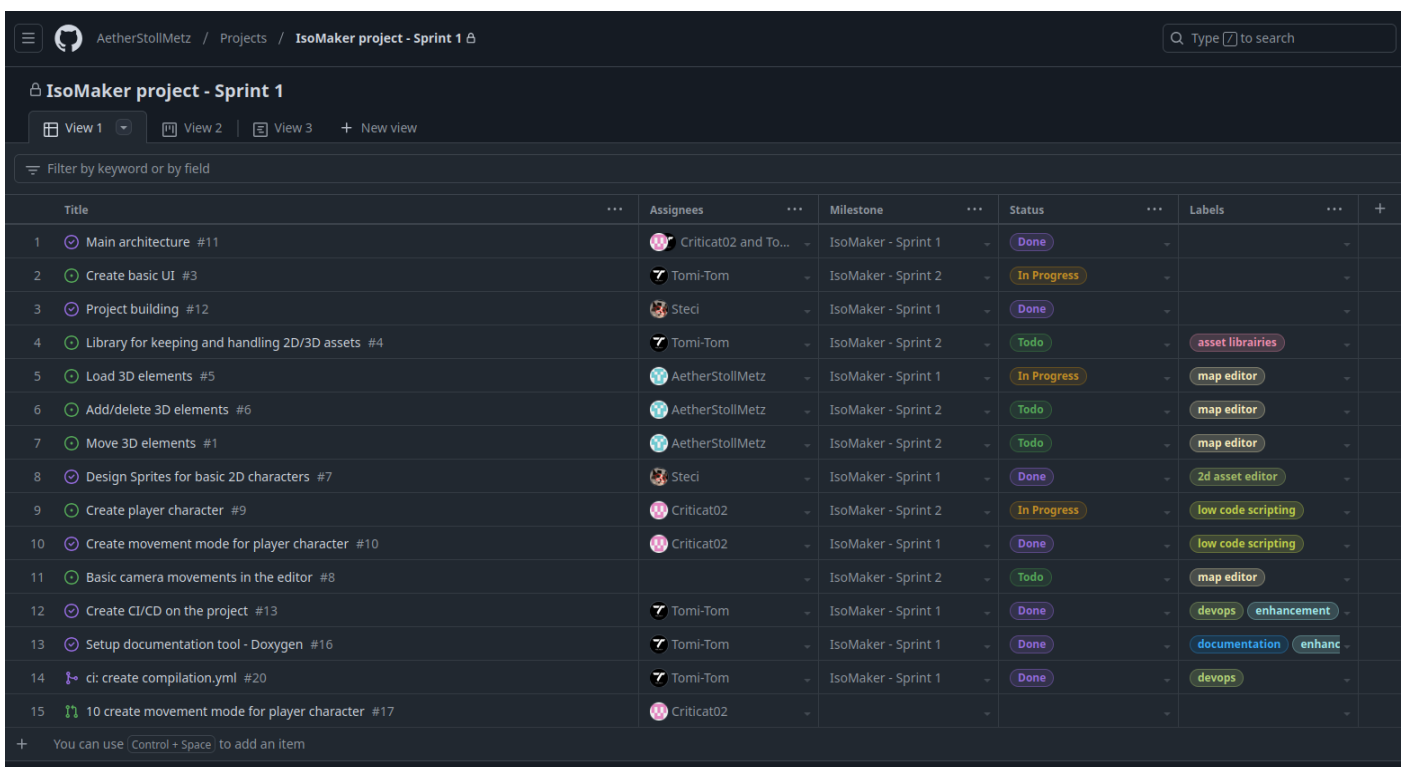
- **Clarity:** Makes it easier to understand what a commit does at a glance.
- **Automation:** Supports tools for generating changelogs and release notes.
- **Consistency:** Encourages contributors to document changes uniformly.
-

# HOW TO ADD A FEATURE AND PULL REQUEST 101

## 1. Access the repository's Github Project



## 2. Add an item to the list



Create new issue in AetherStollMetz/IsoMaker

Blank issue in AetherStollMetz/IsoMaker

Add a title \*

New feature

Add a description

Write

Preview

H B I

☰ <> 🔗

☰ ☰ ☰

@ ↗ ↶

Type your description here...

Paste, drop, or click to add files

Assignee

Label

IsoMaker project - Sprint 1

Milestone

☐ Create more

Cancel

Create

# [New feature] #31

Edit🔗🔄⋮✕

OpenAetherStollMetz/IsoMakerPublic

Criticat02 opened now

No description provided.

😊

Add a comment

WritePreview

H B I ↵ < > 🔗 ⌨️ ⌂ 📎 ↶ ✎

Use Markdown to format your comment

Paste, drop, or click to add files

Assignees

No one - Assign yourself

Labels

No labels

Projects

IsoMaker project - Sprint 1

Status No status ▾

Milestone

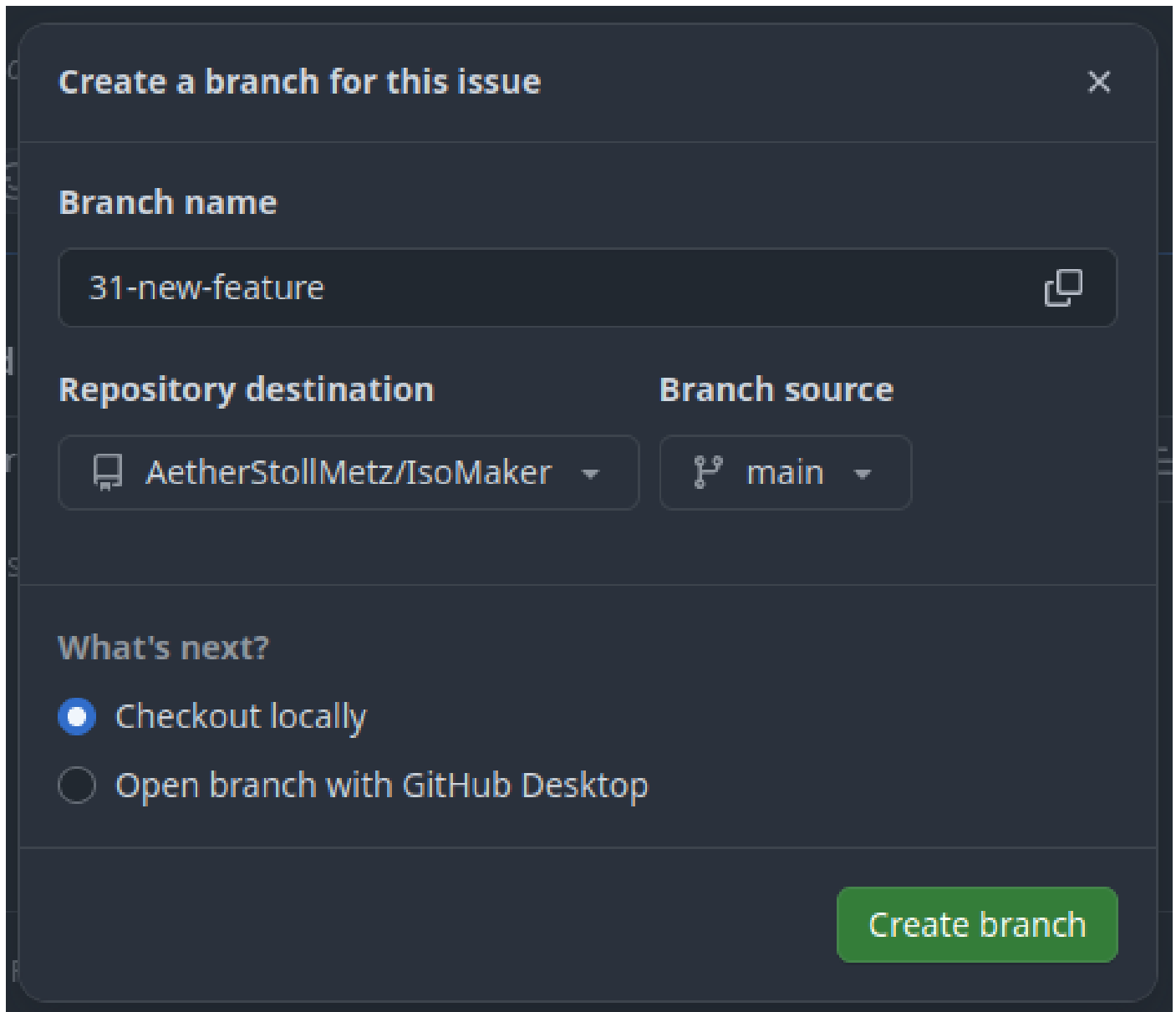
No milestone

Development

Create a branch for this issue or link a pull request.

Close issueComment

5. You may want to make the source branch different from main, otherwise leave everything as is and create branch



The screenshot shows a dark-themed dialog box titled "Create a branch for this issue" with a close button (X) in the top right corner. The dialog is divided into several sections. The first section, "Branch name", contains a text input field with the value "31-new-feature" and a copy icon to its right. The second section is split into two parts: "Repository destination" with a dropdown menu showing "AetherStollMetz/IsoMaker" and a "Branch source" dropdown menu showing "main". The third section, "What's next?", contains two radio button options: "Checkout locally" (which is selected) and "Open branch with GitHub Desktop". At the bottom right of the dialog is a green button labeled "Create branch".

Create a branch for this issue

Branch name

31-new-feature

Repository destination

AetherStollMetz/IsoMaker

Branch source

main

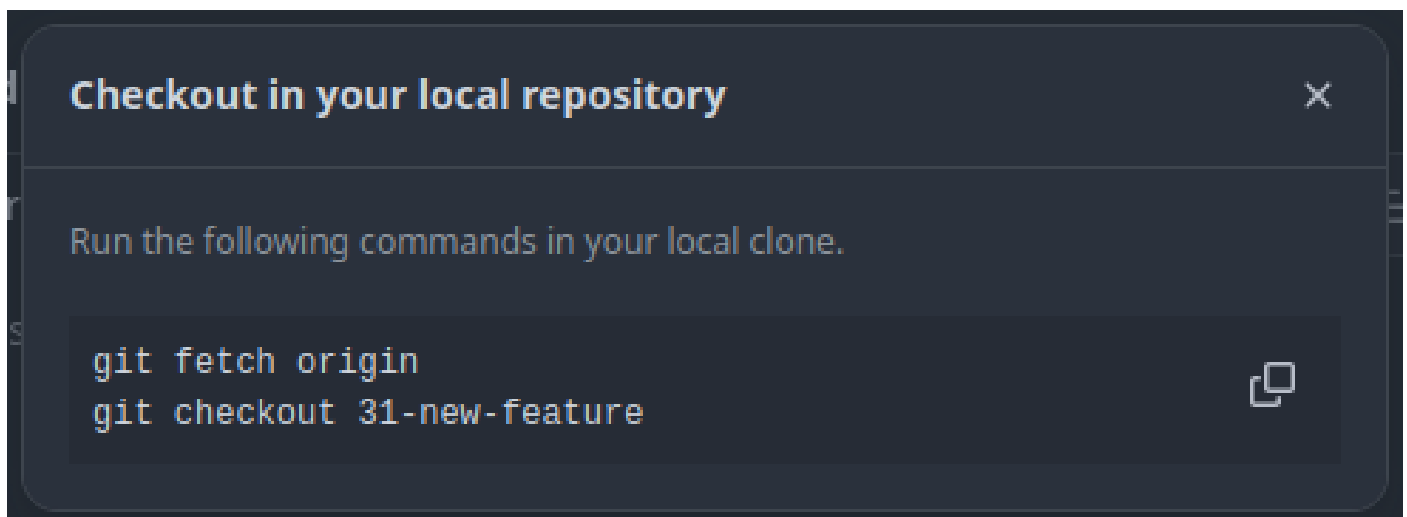
What's next?

☒ Checkout locally

☐ Open branch with GitHub Desktop

Create branch

6. Check out locally and code your feature (make sure to respect the commit message standard when pushing onto the branch)



The screenshot shows a dark-themed dialog box titled "Checkout in your local repository" with a close button (X) in the top right corner. The dialog contains a text instruction "Run the following commands in your local clone." followed by a code block containing two lines of text: "git fetch origin" and "git checkout 31-new-feature". A copy icon is located to the right of the code block.

Checkout in your local repository

Run the following commands in your local clone.

```
git fetch origin
git checkout 31-new-feature
```