Sprint 2 Retrospective

Team Magenta

CSE 3902 AU22

In hindsight, this should not be a retrospective, but instead be a live documentation of events.

Goal Completion

All Sprint 2 tasks on AzureDevOps have been successfully resolved. The only task remaining, the level loader, is for Sprint 3. Tasks that, in hindsight, should’ve been added to the task board for this Sprint are instead added to the Backlog.

Problems

-There are various issues with the code that resulted from a lack of planning and communication. These problems will be added to the backlog, and communication and planning will be improved for the next Sprint.

-Communication problems stem mainly from a misunderstanding of terminology, and strangely named files and folders in the repository. Should rectify as soon as possible.

-clarification of the term “sprite”.

Backlog and Possible Improvements

As stated earlier, despite the smelly code, functionality is retained for the Sprint 1 specifications. However significant refactoring is required.

-Reminder to add concise comments for all classes, including retroactively.

~~-All Updateable game objects~~ **~~must~~** ~~have their Update() methods called~~ **~~every~~** ~~Game update~~

~~-synchronizes cooldowns and in-game time across all updateable objects~~

~~-as they are all being ticked simultaneously at the same rate~~

~~-required for collisions in Sprint 3 :)~~

-Organize and rename files and folders in the repository.

-logically the Updateables folder should include all updateable gameobjects, but does not.

-rename and redistribute Updateables into the various subsets of Entity types, relocate Draw-related classes from Updateables.

-ConcreteSprite and related should be renamed to Entity.

-DrawSprite should be renamed to just Sprite

-Implement a GameObject class that maintains Updateables and Drawables lists.

-Sprint 3 may require a Collidibles list.

-Unify the DrawSprite and DrawStaticSprite classes?

-having a specialized DrawSprite class just for drawing a single texture seems extraneous.

-DRY

-SpriteFactory should be returning Sprite (DrawSprtie) objects

-is currently returning Entities (ConcreteSprite)

-so technically, there is no SpriteFactory

-Problems with the Command Design Pattern.

-controllers are for executing Commands in response to user input.

-despite the name they do not necessarily control game objects in the same sense.

-there should only be one controller, KeyboardController since it is the only means of input.

-all Commands, including ones that control enemies and change tile states live in KeyboardController.

-relocate AI logic from EnemyController into the enemies’ objects themselves.

-the AI belongs to that enemy, and is created and deleted with that enemy.

-this makes it simpler to delete by calling GameObject.

-there should not be an items and tiles controller.

-items are game objects that are updated by GameObject.

-tiles don’t update.

-TileFactory, EnemyFactory, and ItemFactory classes for generating all types of game objects.

-Improve the logic that handles moving the characters

-given that updateables are updated every game tick, move Command can be simplified significantly to **not utilize states**

-implemented using Vector2 velocity (may be needed for collisions)

-Remove all logic from Command classes

-move logic to the classes that the particular Command affects

-e.g. if a Command moves Link, Command should call Link.Move() instead of Command performing the logic for Link’s motion

-Link then has a Move method that moves Link depending on parameters passed by Command

-**reduce coupling**