Game Development - Assignment 2

Overview

Building on top of the previous assignment, we are going to add enemies with pathfinding and options to load / save the state of the game.

Content

Expanding the platformer from the previous assignment, we need to add the following features:

- **Walking enemy type** that can **pathfind** the player. It is not needed that the enemy can jump (although is encouraged) but it should detect that it can reach the player by normal walking and falling down to other platforms (at least 2 enemies instances in the game)
- **Flying enemy type** that can **pathfind** the player, avoiding non-walkable areas. (at least 2 enemies instances in the game)

  (!) The paths from the enemies to the player must be visualised in debug mode. It must show in real time the path that is calculated in each frame from the enemy to the player.

- **Load/Save** the state of the game.
    - Save the Player position and enemies position (Use key F5 to Save and F6 to load)
        - Consider other states like if an enemy has been killed cannot reappear
    - Checkpoint autosave with feedback on passing (Respawn to the last checkpoint after dying). Checkpoint must be defined in the tiled map.
- **Destroy enemies**: jumping onto them, shooting or any other creative way to destroy an enemy, *Optional**:** Skills with cooldown*
- ***Audio feedback***
  Add audio fx to the player actions, killing an enemy, autosave, ….

Entity System: Enemies must hereby from a base Entity class and an EntityManager class must manage them (Initialize, include in a list, Update, Draw, CleanUp…)
Enemies behaviour: Enemies have a range of perception and will be idle until the player is close by.

NOTE: Any remarkable addition to the game beside the proposed ones could be evaluated with extra points to the Assignment mark.

Submission Rules

Each team MUST upload to the task *"Assignment2"* on the online campus the following:

- **release build** as a **zip** file
- The URL of the GitHub Project. The build **MUST ALSO** be published in the **Release section of the project's GitHub page**. The source code will be reviewed from the Release version.
- **A one-page document describing how the work has been organized between team members**. If you use any tool like Trello or Hackaplan (recommended) included the link.
- (!) In the release section, include a brief description of the game and a description of how to play.

Release folder structure and naming conventions:

```
 > Team_Name-Platformer-Beta.zip           // Game directory zipped

      Assets                               // Assets directory, it could contain
                                           // multiple sub-dirs and files
                                           // Assets license files must be near
                                           // the asset file
      Game.exe                             // Main binary for the game (release)
      config.xml                           // Configuration file
      Xxx.dll                              // ONLY required DLLs to run the game
      LICENSE                              // Game license file
      README.md                            // Game detailed info and instructions
```

```
NOTE: GitHub release MUST contain detailed information on the current release
(new features, improvements...)
```

Submission **will not** be accepted for grading in case:

- It is not delivered on time
- Build is malformed (included not used files or code not compiled in **Release** mode)
- Build is not available in the GitHub Release system

Once the delivery is accepted, the **grading criteria** is:

- **(70%) Features:** Evaluation will consider all features completed from the checklist and all the additional gameplay elements and state of completeness.
- **(15%) Coding and project organization:** Code structure, comments and documentation, naming (classes, functions, variables)
- **(10%) Team work.** The GitHub contribution will be reviewed, so make sure contributions are made from separate users. Also, a one-page description of how work was organized between team members must be submitted.
- **(5%) Follow-up submission rules**: Release publication, deliver on time, debug features

NOTE: In case of a great imbalance in work between team members, teacher can decide to downgrade an individual score.

DEBUG keys

Game should include a set of DEBUG options enabled with the following keys:

- **H** Shows / hides a help menu showing the keys
- **F1/F2** Start from the first/second level
- **F3** Start from the beginning of the current level
- **F5** Save the current game state
- **F6** Load the previous state *(even across levels)*
- **F9** View colliders / logic / **paths**
- **F10** God Mode *(fly around, cannot be killed)*
- **F11** Enable/Disable FPS cap to 30

Helpful Links

- Adapting A* to platformers