Game Development - Assignment 1

Overview

The goal is having a simple platformer with a main character with the follow-up camera, a map that is loaded from a tmx file (Tiled), and colliders that are created dynamically from metadata in the map. The physics of the character should be calculated using the Box2D engine.
We are going to optimize the rendering by adding options to control the frame rate and normalize the movement with delta time.

Content

Create a simple platformer game with the following features:

- Design a map based on Tiled TMX map and the functions to load and draw (orthographic)
    - The length of the map is according to the difficulty, but it should not be too simple
    - You can use open source tile sets to create the map
    - The game should support at least one level, a second level is optional, in any case, the process of loading and unloading the levels should work well to avoid memory leaks.

- Platform colliders have to be defined as a layer in the map and are created dynamically at loading time using Box2D.

- Player movement and jumping using Box2D physics.
    - The Player must hereby from a base Entity class and an EntityManager class must manage it (Initialize, include in a list, Update, Draw, CleanUp…)
    - Player walk/jump/die animations
    - Player controls (keyboard):
        - WASD keys for movement
        - SPACE for jump

- Config file loading - read modules and entity parameters from external file
    - Player parameters (initial position, speed, tiles)
    - Animations sets
    - Texture and audio paths
    - (!) The code should not include hard-coded paths

- Game should be capped to **stable 60 frames** per second without vsync.
    - (!) You must set the window title to show the information:
      FPS / Avg. FPS / Last-frame MS / Vsync: on/off

- The game should have all its movement normalized using **dt (deltaTime)**, so in slow/fast machines it would keep the same movement speed.
  (!) To test, Set different frcap values in config.xml (e.g. 16, 32, 64) and all the elements should move at the same speed.

The level should **not** have any UI or enemies. Still, it should require some skill to complete it. The most common way to achieve it is to add fall pits where the player can fall if it does not jump with good timing, but it could be any other challenge (for inspiration, check Super Meat Boy). Make sure jumping is smooth and can go through platforms when jumping from below. Also, make sure you have a good camera.

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-Alpha.zip          // Game directory zipped
    Output                                // Game directory
        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
        save_game.xml                     // Save game
        Xxx.dll                           // ONLY required DLLs to run the game
    LICENSE                               // Game license file
    README.md                             // Game detailed info
```

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:

- **F1/F2** Start from the first/second level
- **F3** Start from the beginning of the current level
- **F9** Visualize the colliders / logic
- **F10** God Mode *(fly around, cannot be killed)*
- ***F11*** *Enable/Disable FPS cap to 30*

Helpful Links

- **The guide to implementing 2D platformers**
- Open Game Art
- Kenney • Assets
- GameDev Market
- Free Game Assets