Game Development - Assignment 3

Overview

Building on top of our previous assignment, we are going to add collectibles, checkpoints, and several GUI elements to the game: a full Main Menu on Title Screen, an Options Menu and an In-game HUD.
We are going to optimize the rendering by adding options to control the frame rate and normalize the movement with delta time.

Content

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

- **(25%) New Game Features**
  - Game items to recover health / lives
  - Game collectible items to accumulate points: coins, stars...
  - Checkpoint autosave with feedback on passing
  - Map checkpoints teleportation (Can move you to another location in the same level or to another level)

- **(20%) GUI: Title Screen Main Menu**
  It must include the following **GuiButtons**:
  - **[PLAY]**: **Smooth** transition to the gameplay screen.
  - **[CONTINUE]**: **Only enabled** if there is a **saved game**. It must transition to the last saved game. Use a different visible state for **disabled**!
  - **[SETTINGS]**: Open **options menu** (in the same screen). Including the following options:
    - Adjust **music volume**, with a **GuiSlider.**
    - Adjust **fx volume**, with a **GuiSlider**.
    - Toggle **fullscreen mode**, with a **GuiCheckBox**.
    - Toggle **VSync**, with a **GuiCheckBox**.
  - **[CREDITS]:** Open a credits panel where you can read the authors and the license.
  - **[EXIT]**: Quit the game.
  - ***NOTE: Buttons must be responsive and include:***
    - Visible state change on **mouse-hover (FOCUSED)**
    - Visible state change on **mouse-pressed (PRESSED)**
    - **Generate an OnMouseClick event to be processed**
    - **Audio feedback** on mouse-hover and mouse-click.

- **(20%) GUI: Gameplay Screen HUD**
  In-game GUI must include the following elements:
  - **Player lives** / **Health bar**: The game will cycle from the beginning of each level until the player consumes all its lifes / health bar. At that point, the game must transition to the ending screen or title screen.
  - **Coins** / **Collectibles:** There must be some sort of coins or collectibles that the player must be able to collect throughout the game. The HUD must show the amount collected so far. Their effect into the game is up to you. They must be respected when saving/loading the game state.
  - **Timer:** It should contain the time so far accumulated from the player while playing the level or the remaining time to finish the level. The player must die if it doesn't arrive to the finish line on time.
  - [Optional] **Score/High-score:** Some scoring mechanism (based on time/collectibles) can be considered and stored to make the game more appealing to players.

- **(5%) GUI: Gameplay Screen Pause Menu**

  Pressing ESCAPE triggers an **in-game Pause Menu** with the following options **GuiButtons**:
  - **[RESUME]:** Continue playing the game after displaying the menu.
  - **[SETTINGS]:** Shows settings menu (same one from Title Screen).
  - **[BACK to TITLE]**: Returns to title screen.
  - **[EXIT]:** Quits the game.

- **(20%) Optimization**
  - Game should be capped to **stable 60 frames** per second without vsync.

    (!) You must set the window title to show: *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 teh elements should move at the same speed.
  - Profile the game using Optick profiler. Include a screenshot of the result of your profiling in the deliverable.

- **(10%) Follow-up submission rules**: Release publication, code organization, deliver on time, debug features

Bonus: Additional features to recover grade from assignments 1 and 2

- The implementation of features that were not working in previous assignments can recover 50% of the evaluation of the grade of the feature (example: if the feature was 10% of the grade, it can recover 5%). The assignment grade will be updated. The reimplemented features must be listed.

NOTE 1: *The GUI should NOT be static! It must attract the player's attention when something important happens, i.e., animated coins increment, animated lives update, sounds… check your favorite platformers games for reference.*

NOTE 2: All the elements in the game (Player, Enemies, Props, Coins / Collectibles) must hereby from a base Entity class and an EntityManager class must manage them (Initialize, include in a list, Update, Draw, CleanUp…)

NOTE 3: All elements must be defined and loaded from the config.xml or from the map file (tmx) (coins, collectibles, checkpoints)

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
- **F5** Save the current game state
- **F6** Load the previous state *(even across levels)*
- **F7** Move between the different checkpoints
- **F8 View GUI bounds rectangles and state in different colors**
- **F9** View colliders / logic / **pathfinding paths**
- **F10** God Mode *(fly around, cannot be killed)*
- **F11** Enable/Disable FPS cap to the parameter frcap in the config file

Submission Rules

Each team MUST upload their **release build** as a **zip** file to the folder *"Assignment3"* on the campus website **no later than Sunday January the 29th at 23:59** *(folder closes automatically)*. If more than one file is uploaded, only the last one will be evaluated.

The build **MUST ALSO** be published in the **Release section of the project's GitHub page**, source code will be reviewed from the Release version.

Release folder structure and naming conventions:

```
> Team_Name-Platformer-Gold.zip      // Game 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**
    - **Delivered files are wrongly named**
    - **Not compiled in Release mode**
    - **Includes DEBUG temporal files**
- **Build is not available in the GitHub Release system**
- **Game crashes while testing**

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

- **100% Checklist points:** Evaluation will consider all points completed from the checklist and all the additional gameplay elements and state of polishment.

Grades could be **downgraded to 50%** if not fulfilling the following rules:

- **Assets must follow the Files/Directories naming conventions.**
- **Code must follow the Code Naming Conventions.**
    - Be very careful to use TABs instead of 4 spaces!
    - Be very careful to use only one space between *DataType field;*
- **Directories organization is not the expected one**
- **Release includes assets not used in game**

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