Design Plan

Genre

We are setting out to build a turn based strategy game engine.  The inspiration for this came from turn-based “war-like” games in which one player controls an army of soldiers against a computer’s army.  Some examples of such games (with gameplay videos for clarity) are as follows:  
  
Fire Emblem (probably already better than what our final product will be…):  
<https://www.youtube.com/watch?v=SAQpa4xYB_o>

Fire Emblem: Radiant Dawn (much prettier version of the same idea): <https://www.youtube.com/watch?v=UzGcCBnWicY&list=PLF007A82BEC04F667&t=1110>

X-Com: Enemy Unknown (even cooler!):  
<https://www.youtube.com/watch?v=YY8Rkyps_PU&t=380>

The user will not be limited by our authoring environment to making games of this exact form.  Using our engine, users will be able to build games in which a primary player moves units / characters across a grid-like map on his / her turn and all other “players” (computer AI + possibly other human players) are allowed to move their units / characters before the player can move again.  This would ideally be flexible allow a creative mind to create variants of board games, puzzle games, and more in addition to the “obvious” route of making a turn-based strategy game.

Design Goals

-Allow user to toggle between game mode and design mode at will

-Extra credit portions: #s 1, 2 (see above), and 7

Design Specifications

1.Authoring Environment (Teddy, Andy)

Character ( currentLife, maxLife, collisionID, name, location, imageName, power, defense, movement/speed, attack range) // may extend environmentObjects()

Player() extends character

Enemies() extends character

Environment Objects(collisionID, name, location, imageName)

Terrain

Objectives(check points)

2.GameEngine Interface (Grace, Xin, Matt)

(swing stuff - panels, windows, buttons)

* Preferences: size of grid
* Panels: JGEngine, ButtonPanel, GameObjectSelector, State, EnvironmentObjects
* Drag and Drop animation
* Menu bars
* Class ButtonPanel which manages all buttons, e.g. Play, Save, Load, Preferences

3. Controller

- it’s a class that connects GameEngine, Authoring Environment with Game Data

- Fields: Mode (Edit/Play)

- Methods: put objects on the GameEngine, call methods in utility class to load and save, AI algorithm to compete with player

4.Game Data (TC, Thanh-Ha)

* Save GameData(Object [ ][ ] map), utility class to serialize game data into files
* Load GameData(): utility class to load files into map
* Store all game objects and preferences: class Slide (as one level) which stores all current objects and preferences, class SlideManager which stores all slides and open APIs to access active one

5.Game Player ( Evan, Kanchan)

* Module which handles gameplay when program is in “Play” mode (as determined in Controller)
* Fields:
  + Score:  keep record of current/high scores of player
  + Turn:  track which player is eligible to move at any time
* Methods:
  + doFrame():  execute movements, animations, and actions on the playing field
  + checkCollisions():  handle object collisions and remove objects when necessary
  + Listen for and respond to input from user by calling appropriate methods from Game Engine
  + Check for level completion and increment levels when appropriate

Sample Code

(Apparently this isn’t supposed to be actual code; rather it should be a description of 2-3 example games that can be designed using our game design environment, and pertinent data files)

Examples games (probably can be expanded):

* Kill everything in the world to win/advance
* Kill a boss to win/advance
* Reach a certain capture point to win/advance
* Collect a certain amount of objects before time runs out, etc.
* Survive for a certain number of turns
* Math grid games like 2048...it’ll ruin lives

Design Alternatives