Game Development TRS - Revaluation Exam - January 31st 2017

YOUR FULL NAME: **Solution provided by the teacher**

- You have 2 hours to complete the assignment.
- Only valid text will be the one inside each box, everything else will be ignored by the teacher

1.**(3 points)** Adapt the A* algorithm to work on a map that can wraparound. This means that leaving to the left should be connecting you to the right (we assume the is **no** wraparound top-bottom). Describe the pathfinding phases that would be affected and how. E. g.:

	X	Start									
X	X								Dest	X	X

```
while the open list is not empty
  find the node with the least f on the open list, call it "q"
  pop q off the open list
  generate q's 8 successors and set their parents to q
  for each successor
    if successor is the goal, stop the search
    successor.g = q.g + distance between successor and q
    successor.h = distance from goal to successor
    successor.f = successor.g + successor.h
    if a node with the same position as successor is in the OPEN list which has a lower f than successor, skip this successor
    if a node with the same position as successor is in the CLOSED list which has a lower f than successor, skip this successor
    otherwise, add the node to the open list
  end
  push q on the closed list
end
```

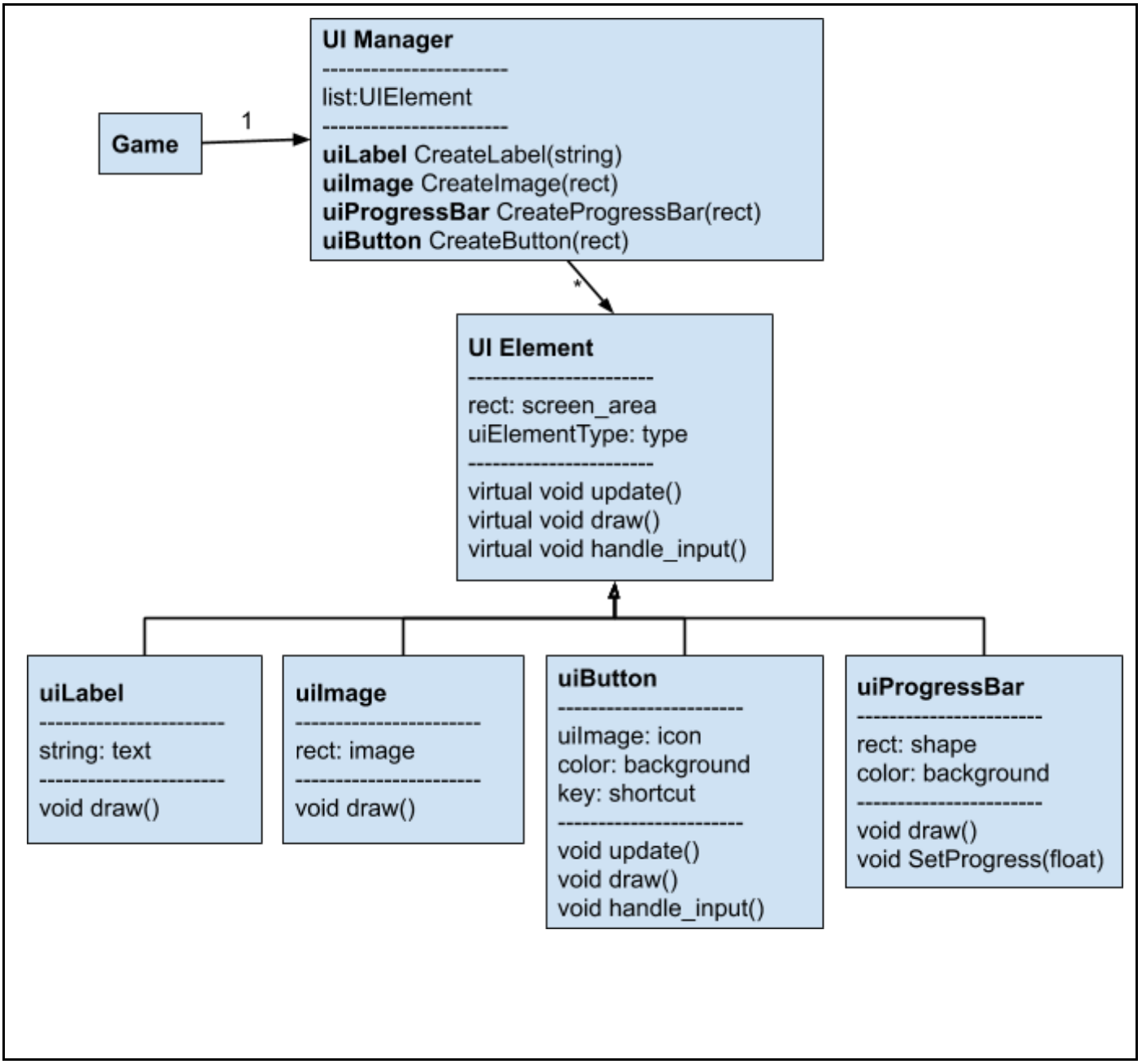
- We would affect phase for opening adjacent squares just by taking in account the wraparound:
  - Given x_offset the current offset to find adjacent squares on x axis:
    - **if** x_offset < 0, then x_offset = *map_width*
    - **if** x_offset >= *map_width* then x_offset = 0

- Then distance calculations would be changed to simulate a world that has copies from itself on the right and on the left, and take the smaller distance:
  - distance from square to goal: min(A to B, A to B + map_width, A to B - map_width)

	A1						A						A2				
				B1						B						B2	

- In the above example, to calculate the distance from A to B, we would take the smaller distance out of three: A to B, A to B1, A to B2

2. **(3 points)** Design the UML of a GUI system that could handle all elements in this image:



3. **(2 points)** If we have a game where the logic uses 25 ms with vsync turned on (monitor refresh rate of 50 Hz) and our main character moves at 200 pixels per second. How much, in average, is he moving each frame taking in account that we have variable time step ? How much vsync will make the application wait ?

With a monitor at 50 Hz we and vsync turned on we can only operate at 50, 25, 16, 12, 10, … fps

If the game logic is consuming 25 ms it means is running at 40 fps, but vsync will have us wait until we go down to 25 fps, which totals 40 ms. ***This means that vsync will wait for 15 ms.***

*Our main character will be moving with a dt of 1/25 = 0.04 This means that it will move at 200 * 0.04 =* ***8 pixels per second.***

4. **(2 points)** Explain the key concepts behind a Quake-style console. Explain the main steps that need to be taken in order to implement it (assuming we have a functional GUI system in place).

A quake-style console is a tool were we can input commands into our game similar to a traditional command prompt of an operative system. It provides a tool for fast iteration on gameplay elements without closing the application. We can quickly check the current values of different gameplay elements, tweak them and save them for future uses.

The main elements are the commands and the cvars (console variables). The first are for executing specific functionality, like loading a new level, saving to a file or quitting the game. The cvars store values that are relevant to the game, like player HP, damage performed with different attacks and so on.

The console also outputs many debug information that can be scrolled up for fast tweaking/debugging of the game.

In order to implement it, the strategy would be (there are many valid ways, this is what was proposed in class):

- Creating a new module to handle all commands and cvars
- Using the GUI system we create a multiline label and an input box
- The GUI should appear/disappear on pressing a key (normally tilde, left of 1)
- We redirect all LOG entries to the multiline label
- We make sure there is a way to scroll up/down on the multiline label (normally with mouse scroll using the third mouse button)
- We create a small class to represent the concept of command and cvar
- We create method to add new commands and cvars with callbacks for each module to handle their specific implementation
- Optionally we can save the historic of commands and have autocomplete feature in the input box

Use this page for your own notes. You cannot use any other page. Do not remove this page from the set.