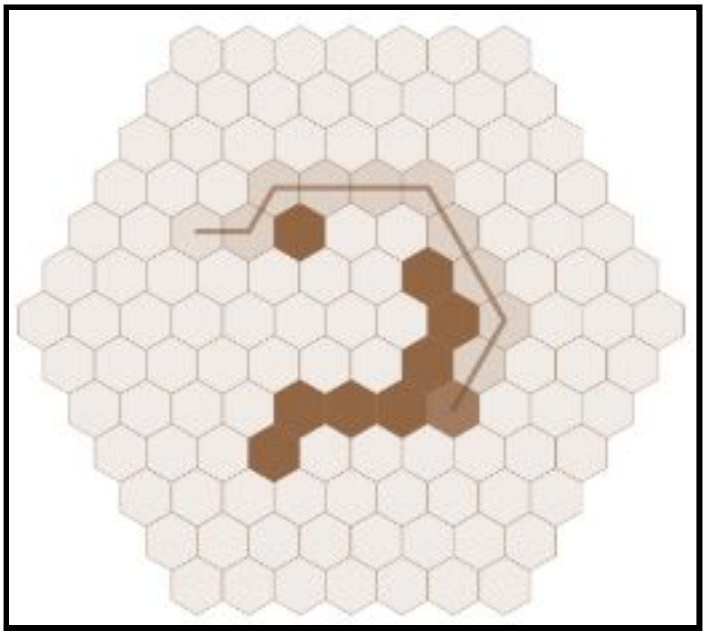
Game Development BCN - Final Examination - January 16th 2018

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)** Explain would you adapt the A* algorithm to hexagonal grids. What would change, what would stay the same and **why**.



The core of the algorithm would stay the same since A* does not presume any specific amount of children per node in the graph.
Things to take in account:
- Now we have six neighbors all at the same distance from each other.
- We need to calculate the distance between two hex nodes. We have two options:
  - We have a translation function from hex coordinates to world coordinates, then we just use normal distance equation.
  - We implement a “hex distance” function that returns the amount of hexes between two other hexes in hex space.

2. **(2 points)** If we have a game where the logic runs at 100 frames per second with vsync turned on (monitor refresh rate of 75 Hz) and our main character moves at 150 pixels per second. Taking in account that we have variable time step, how many pixels the character moves every frame ? And if we suddenly drop our logic frame rate to 32 ? **Elaborate** your answer.

Even if our logic frame rate is at 100 FPS, we will go down to 75 FPS with vsync activated due to time wasted in vsync. This means that our average frame will have a time differential of 0.0133 seconds

*Knowing this, in an average frame the main character will move 150 * 0.0133 =* ***~2 pixels per frame***

*If we drop to 32 logic FPS, we would go down to 25 FPS due to vsync, so our dt would be 0.04. That means that our main character will move 150 * 0.04 =* ***6 pixels per frame***

3. **(2 points)** Explain the concept of an **Class Factory** and how it applies to an Entity System. Write a simple example in C++ were the only types of entities are "*player*" and "*npc*".

A Class Factory hides the complexity of creating different types of classes that inherit from the same type. The factory assures to return a pointer to the base class that can be safely cast to the child class type.

An example for a simple Entity system would be:

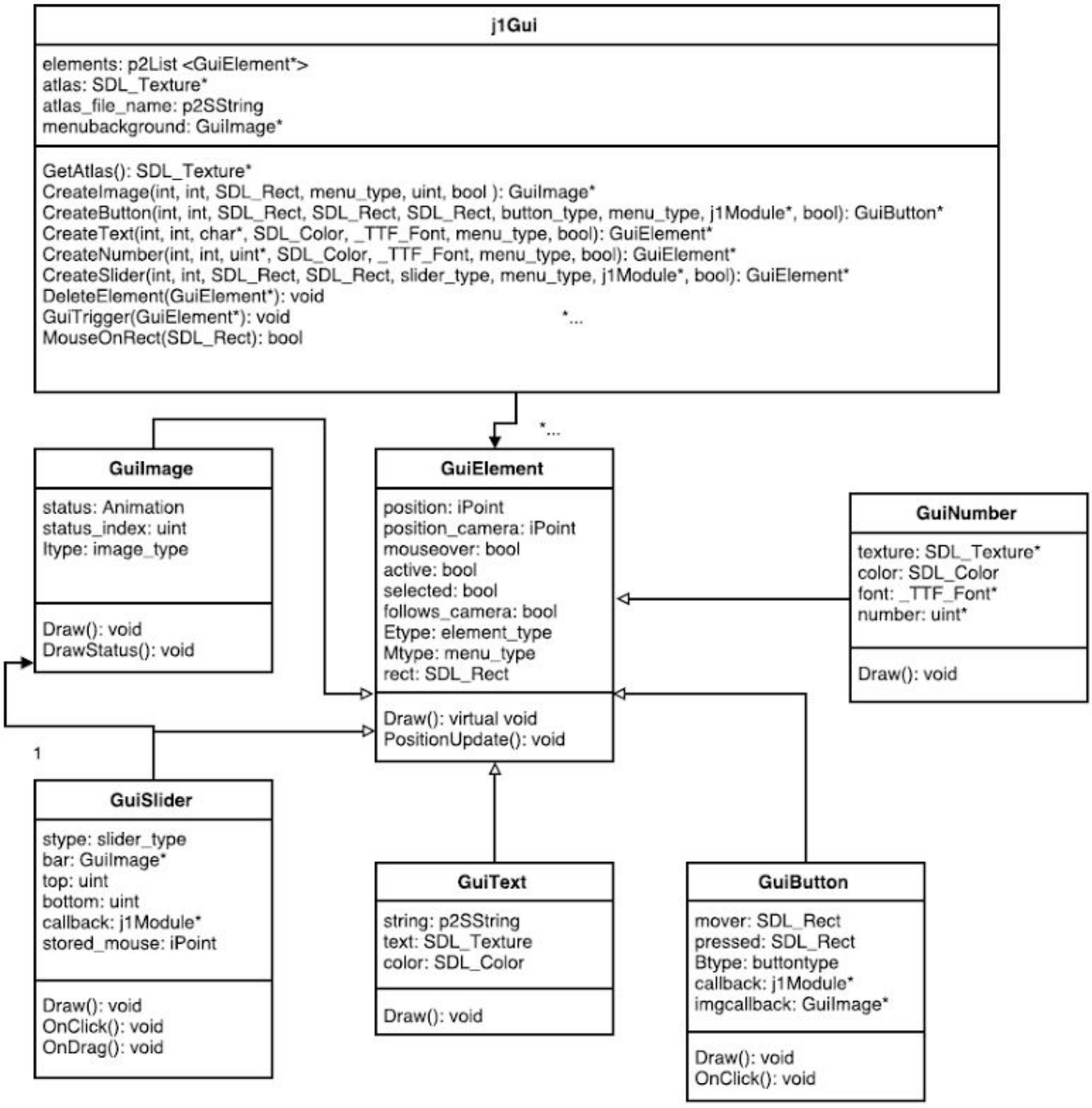
```
Entity* EntityManager::CreateEntity(Entity::Types type)
{
	Entity* ret = nullptr;
	switch (type) {
		case Entity::Types::npc:	ret = new NPC();	break;
		case Entity::Types::player: ret = new Player();  break;
	}

	if (ret != nullptr)
		entities.push_back(ret);

	return ret;
}
```

4. **(3 points)** Given this UML that describes an UI system, point out the mistakes and how you would improve it. Do not create another UML, focus on improving this one. Categorize every mistake with a keyword: **C:** critical, **I:** important and **O:** optional.

Example: *"C: j1Gui::menubackground is hardcoded value, remove it"*



- C: j1Gui::menubackground is hardcoded value, remove it
- O: Simplify GuiElement creation with a Class Factory
- I: CreateText/CreateNumber/CreateSlider return the base class
- C: j1Gui::GuiTrigger is hardcoded value
- I: j1Gui::MouseOnRect could be generalized somewhere else
- O: Add from where it is created j1Gui and how many are there
- C: GuiElement::position_camera is confusing, guess it is the global coordinate. In this case must be a function
- I: GuiElement::mouseover must be a function not a property
- I: it is not clear what means GuiElement::selected, if it is for the tab order, this must be inside j1Gui
- I: not clear what is GuiElement::Mtype, but looks like it can be removed and generalized
- I: GuiElement::rect is only applicable to images that cut from the atlas, won't be used in GuiText and GuiNumber. Push it to child classes.
- O: GuiElement::PositionUpdate would be called from jGui, but there is not a method defined for it (Update() ?)
- I: GuiImage::status is redundant with GuiElement::rect
- O: GuiImage status is the animation index ? if so the name is confusing
- I: GuiSlider: the thumb for the slider must be another GuiImage*
- I: GuiImage::callback can be generalized in GuiElement
- I: GuiImage::OnDrag can be generalized in GuiElement, as for the tumb can be limited to one axis between two coordinates
- I: GuiButton::callback can be generalized in GuiElement
- I: GuiButton::imgcallback is not clear what it does (?)
- I: GuiText lacks font (but GuiNumber has one)
- I: Remove GuiNumber, is just a case for GuiText

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