INFO6044 – Game Engine Frameworks & Patterns

Midterm Exam – Friday, October 28th, 2016

Instructor: Michael Feeney

The exam format:

- You may use any resources you feel are necessary to complete the exam, but you are to answer the questions **on your own**. I will be looking for plagiarism (i.e. copying) very carefully. There is *no possible way* that the specific code to answer these questions, or the output to the screen, would be very similar to the look of another student's code. Remember, this is a test and there are very clear policies about cheating on tests.
    - http://www.fanshawec.ca/admissions/registrars-office/policies/cheating-policy
    - http://www.fanshawec.ca/sites/default/files/assets/Ombuds/cheating_flowchart.pdf
- It is an "open book" exam. You have access to anything you book or internet resource you'd like
- The questions are ***NOT*** of equal weight. The exam has **five (5)** questions and **five (5)** pages.
- Your solution can be either graphical or console based (or graphical + console based if that's helpful).
- **CLEARLY** indicate which answer goes to which question. My suggestion is that you place each answer in its own folder, named "Question_01", "Question_02" and so on (or something equally clear). Another option is to create a Visual Studio solution and add a number of projects – one per question – to it. If I can't make heads or tails of what question is what, I probably won't even mark it.
- Do ***NOT*** do some clever "*oh, you just have to comment/uncomment this block of code*" nonsense. However, if the questions ***CLEARLY AND OBVIOUSLY*** build on each other, you may combine them (like if one question places objects, then the next one moves objects around with the keys) – even so, **MAKE IT 100% CLEAR** to me what questions the solution is attempting to answer.
- Place any written ("essay" or short answer) answers into a Word, RTF, or text file. Again, *clearly* indicate which question you are answering.
- If you are combining answers (which is likely), please indicate this with a "readme" file or some note (*not* buried in the source code somewhere).
- For applications: if it doesn't build and run, *it's like you didn't answer it*. I'll correct trivial, obvious problems (like you clearly missed a semicolon, etc.), but you need to be sure that it compiles and/or runs.
- You have until **11:59 PM** on **Friday, October 28th** to submit all your files to the appropriate drop box on Fanshawe Online.

  NOTE: Although this may "look and feel" like a project, it isn't, it's an **exam**, so there is **no concept of "late marks**"; if you don't submit your files the time the drop box closes, you don't get any marks at all.

Please don't be late submitting.

(Also be **SURE** that you are actually submitting the correct files)

- Your solution may **not** contain any third party "core C++" libraries (like boost) or C++11 "auto" feature. If it has either of these things, the question(s) will not be marked (because it won't build). You many have other "utility" libraries, like ones to load textures, models, sounds, etc.

- When ready to submit, please delete all the "extra" Visual Studio files before zipping it up (remember this is C++, so all I really need is the .h and .cpp files, right?), like the "Debug" and "Release" folders with the "obj" files, as well as the intellisense file

- **If the solution does not build (and run), I will not mark it** (so you will receive zero on questions that can't be built and/or won't run). When I say "run", I'm not speaking about some, random, unforeseen bug, but rather something that you should have obviously dealt with, like memory exceptions, etc.

- Unless otherwise indicated, all these solutions assume that you are creating/using a C++ project using Visual Studio 2008 through 2015 using the OpenGL 4.x API (with freeGLUT, GLEW, and GLM).

- I'll be at the school from 8:00 until about 3:00 PM. If anyone is in the classroom, I'll stay there until the end of the usual class time. After that, you can reach me through e-mail (mfeeney@fanshawec.ca), by calling the school, or by calling my cell: 519-494-7569 (I'll be on my way to Windsor for the ACM programming competition).

The Questions:

"It's a dog eat dog world out there..."

You have been asked to create a basic simulation of a small ecosystem, mainly to so see if it's "stabilizes" over time.

There are various animals and plants scattered through a world. The interplay between these creatures is based on a number of simple rules.

The only mathematical contract you will need is the distance between two points. This can be done using Pythagorean theorem, or using a "distance" or "length" method/function a library like glm.

$$distance = \sqrt{((x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2)}$$

Fruits/Plants

Type	**Calories**	**Spawn time**	**Time to spoil**
Strawberry	100	2 – 5 seconds	20 seconds
Pineapple	1000	5 – 15.0 seconds	75 seconds
Watermelon	10,000	10 – 25.0 seconds	60 seconds

Animals:

Type	**Starting Calories**	**Food preference:**	**Time to reproduce**
Bunny	10,000	Strawberry, Pineapple, and Watermelon, evenly	60 seconds
Fox	25,000	#1: Bunny #2: Pineapple	75 seconds
Tiger	500,000	#1: Bunny #2: Fox #3: Strawberry	200 seconds

The "world" is a square area 1000x1000 meters.

There are 500 plants randomly scattered throughout the world, with equal likelihood. In other words, there should be the approximately the same number of Strawberry, Pineapple, and Watermelon plants somewhat evenly spread about the world.

Plants can either have a fruit or not. If they *don't* have a fruit, then they will grow another one at some point within the "spawn time". If they *do* have a fruit, and it isn't eaten by something, it will "spoil" (and disappear) in the "time to spoil" time. All plants start without fruit.

For example, at the start of the simulation, all the Strawberry plants *don't* have fruits. Each plant will grow (spawn) a fruit some random time between 2-5 seconds later. If something eats the fruit (explained later), then 2-5 seconds later another fruit will appear. If nothing eats the fruit, 20 seconds later, the fruit disappears and 2-5 seconds later another fruit will appear.

Start with 20 of each animal, randomly scattered through the world. Animals have particular food they eat, and may have a preference or not. If there is a preference, they will attempt to eat the higher preference food if it's available. If not, they choose the next favorite. For example, a Fox will try to eat any Bunny that's around, but if there a no bunnies, they will eat Pineapple plants.

Animals always eat the closest food. All animals move at the same speed, so the closest animal to a particular food will always get it. If there is a "chain" or animals and food, everything in the chain, except the "top" animal, is eaten. For example, if a Bunny attempts to eat a Strawberry, while at the same time a Fox is eating *that* Bunny, *and* a Tiger is attempting to eat *that* particular Fox, then the Strawberry, the Bunny, and the Fox will be "eaten" (disappear), leaving only the Tiger.**

**NOTE: You *don't* need to reinforce this "rule" in your code, but if you start with the "lower" animals on the food chain, and work "up", this should happen automatically.

You are to simulate the world every second, updating the animals and plants. At every "tick":

- The animal loses 0.5% of their "calories" (aka "health") relative the starting calories they could have. So a Bunny, being 10,000 calories, would lose 50 calories every second. Once there are no calories left, the animal is "dead". Note that *unlike* the fruit, the animals bodies remain until eaten, but with 5% of their maximum calories (i.e. a *dead* Bunny can be eaten for 500 calories).

- When the animal gets to 50% of its starting calories (for the Bunny, this would be 5,000), it will start looking for food, based on preference, and pick the closest appropriate food choice. It will move immediately, and instantly, to that food and eat it, absorbing those calories. Animals and plants that are eaten are removed from the simulation (this is when plants start their "re-spawn" countdown). The first animal to reach the food eats it ("first" being the first to be simulated); note that this really won't be an issue as you will be simulating animals one at a time – i.e. the 1st Bunny to eat a Strawberry removes it from the simulation, so there isn't a Strawberry there for the 2nd Bunny to go after. If there isn't any appropriate food, they go hungry (but perhaps other food will spawn in the next moment?).

- Like the plants "spawn time", animals have a countdown to when they can reproduce (mimicking initial maturity and gestation). If the "Time to reproduce" has passed a new animal is spawned, and the countdown is reset. The "child" animal spawns in the same location as the parent.

- The plants use this "tick" to determine when fruits are spawned and when they go bad.

1. (20 marks) Create the "world" and place the 500 plants and 60 animals in it. If you are making a graphical solution, show this will different shapes and/or colours. If it is a console based application, list the location and types of plants, and the location and types of animal.

   Do <u>not</u> implement "behaviour" of the animals and plants at this time.

2. (40 marks) Implement the "behaviour" of the fruit, and show this running for at least 100 seconds. If it is a graphical solution, we should see the fruits appearing (spawning) on the plants, then "going rotten", and re-spawning. If it is a console based solution, choose 10 plants and print out the status/events for the first 100 seconds (something like "Second 6: Strawberry plant 2 grew a strawberry" and "Second 72: Strawberry plant 6 had a fruit go rotten"). I don't want a barrage of thousands of lines of text to plow through, trying to see if your simulation is working or not.

   Do NOT implement the animal behaviour, yet.

3. (40 marks) Implement the "behaviour" of the animals, and demonstrate their calories ("health") going down and them picking their first food. DON'T have them eat anything, yet; once they pick a food, just indicate this and the time.

   If it is a graphical simulation, you can move the animal to the fruit and change the animal model's colour or something.

   if it's a console simulation, only list when the animal fist does this, a suggestion being "Second 24: Tiger 3 is attempting to eat Fox 2". Again, I don't want a giant wave of text to process.

4. (40 marks) Complete the simulation, so that animals that eat plants, and plants that have "rotten" fruit, remove the objects from the simulation.

   If this is graphical, the animals can simply disappear (or fade away, if you're into that sort of graphic "fanciness").

   If this is a console solution, only indicate key events or changes in the animal and plant life. Don't print out hundreds of lines of "Tiger 2 didn't do anything".

   Note that even if you are using a graphical output, you can still print things to the console (since we have access to the console with GFLW).

   If all the animals die, the simulation is over. In both types of application, print a console based summary of this.

5. (40 marks) Add the following animal and plant to the simulation and repeat question 4:

   Fruits/Plants

Type	Calories	Spawn time	Time to spoil
Lettuce	250	1.5 – 7.5 seconds	40 seconds

Animals:

Type	Starting Calories	Food preference:	Time to reproduce
Unicorn	300,000	Pineapples or Lettuce	125 seconds