

**Foundation Degree in Digital Innovation**

Module Name **Advanced Programming**

Year of Study Year 2 (cohort 4)

Assessment **Module Assignment**

Name of Student …………………………………………………

Name of Module Lecturer Alasdair Blackwell

Assignment Weighting 100%

Released Date 2nd July 2019

Submission Date Wednesday 4th September 2019 @ 12 noon

|  |
| --- |
| PLAGIARISM: please refer to the student handbook for the college policy on plagiarism and procedures for correct referencing. Material should not be copied directly from the web or a publication without a correct acknowledgement at the place of insertion of the material. It is not sufficient to include a bibliography reference only. It is also an offence to copy work from another student.  Statement (to be signed by the student before hand-in):  "I confirm that the submitted coursework is my own work and that all material attributed to others (whether published or unpublished) has been clearly identified and fully acknowledged and referred to original sources. I agree that the College has the right to submit my work to the plagiarism detection service Unicheck for originality checks."  Signed…………………………………………………………. |

Where coursework is submitted late and there are no accepted extenuating circumstances it will be penalised in line with the following tariff:

* Submission within 6 working days: a 10% reduction for each working day late down to the 40% pass mark and no further.
* Submission that is late by 7 or more working days: submission refused, mark of 0.

**1st Marker Signature………………………………………Mark………………….Date………**

**Mark after late submission penalty (if applicable)**

**2nd Marker Signature……………………………………...Mark …………………Date ……… (if applicable)**

**Assignment Description**

**Part A (20%)**

1. With the aid of appropriate code examples, describe the difference between the imperative and declarative programming approaches. (5 marks)
2. Explain the distinguishing features of the object-oriented, functional and procedural programming paradigms. For each paradigm, give an example of a system that would benefit from being developed in that style, making reference to the particular paradigm features that would lead to the benefit. (15 marks)

*Your report should be no more than 1-2 sides of A4*

**Part B (10%)**

Draw a UML Class Diagram representing the following elements from the problem domain for a hockey league.

“A hockey league is made up of at least four teams. Each team is composed of 11 to 15 players, and one player captains the team. A team has a name and a record. Players have a number and a position. Teams play games against each other. Each game has a score and a location. Teams are sometimes led by a coach. A coach has a level of accreditation and a number of years of experience, and can coach multiple teams. Coaches and players are people, and people have names and addresses. “

Draw a class diagram for this information.

**Part C (70%)**

**Farm Manager: Main description**

Farm Manager allows gamers to take on the role of a farmer in a world that’s changing faster than the seasons. Climate change, the rise of veganism and the death of Joe Grundy are just three of the outward signs of an urgent need to pivot. But where some see threats, others see opportunity with solar panels, vegan sausage rolls and bio-fuels in high demand.

Farm Manager’s goal is to get people thinking practically about the effects of climate change, and to serve as a contemporary, digital re-imagining of the inner-city school trip to the country to meet the cows.

A developer has kick-started the project but they have been taken ill. You have been hired to finish it off.

**The brief**

There is just three days’ budget remaining. Spending **not more than twenty hours in total**, your job is to progress the project, implementing as many features as you can and documenting your work.

**Basic requirements**

* The project should be coded using P5 as a single-page JavaScript application. Typescript is preferred for the models, whilst JavaScript is OK for your UI components. Any custom development work should be unit tested where necessary.
* Loading the application should present a split-screen with the animated game screen (the farm) on the right and a dashboard on the left. Both of these views should read from the same models.
* The user controls the farmer using the arrow keys. Travelling to a different point on the farm should reveal further information about that part of the farm in the Dashboard. The user can use on-screen controls (form elements such as buttons, inputs, select options and so on) to create game events.
* Your models should be programmed using the object-oriented programming paradigm. You are expected to make use of inheritance, polymorphism, and abstract classes. For your UI components (built with React in the original code base), you are welcome to choose between an object-oriented approach using e.g. React Component classes, or a more functional style e.g. using React Hooks.

**Specific requirements**

**Animals**

User can farm the following animals:

* Cows which yield milk and beef, and eat straw
* Sheep which yield wool and lamb, and eat straw
* Chickens which yield eggs and chicken, and eat corn

*All animals can yield tourism revenue from a field that’s been converted into a petting farm.*

Animals get hungry once they have been milked, sheared, or had their eggs collected. If animals stay hungry for too long, they die. Yields from hungry animals are less than yields from happy animals.

**Crops**

Users can farm the following crops:

* Grass which yields straw
* Wheat which yields bread
* Corn which yields corn

*Grass can also yield green gas in a field that’s been converted into a green gas plant.*

**Fields**

Fields can be used to graze animals or grow crops. Fields left fallow increase in nutrients which leads to higher yields the next time they’re used.

**The Market**

The user is able to visit the market that buys and sells produce and supplies. Market forces are at play, with demand varying based on consumer tastes and trends.

Users are able to purchase solar panels and the equipment needed to convert fields into green gas generators.

Users can also purchase more land to acquire more fields.

**The Weather**

Farming is susceptible to the weather more than any other industry. Connect to a weather API and load different effects based on the response. For example, if it’s sunny your animals might need to drink more water. If it doesn’t ever rain, your crops might dry up and die.

**Logging in, saving state, and extending the farm**

Users should be able to save their progress and come back to it later on the same device. Users should also be able to create a login and access their farm on any device. Users should also be able to create multiple farms.

**Mobile play**

Whilst the game is primarily intended for a desktop user, the client has requested it work on mobile too.

**Fun**

The game should be fun. The client has requested fun animations for animals, crops, trips to the market, and so on.

**Get creative**

The client has been very clear that you are welcome to bring your creativity to this project, as long as what you produce meets the basic requirements and is in line with the main description of the game. You can also delete the code that’s already there and work with alternative JS libraries (such as Angular, Vue.js, or using vanilla JavaScript).

**Marking Scheme**

**Pass Criteria (20%)**

Code base should implement animals, crops, and the market with basic features, as well as demonstrating at least **four of the concepts from categories A-E, outlined below**.

**Merit Criteria (20%)**

Code base should implement the above (pass criteria) plus the weather, green gas, petting farm, and complex market forces, as well as demonstrating at least **another four of the concepts from categories A-E, outlined below**.

**Distinction Criteria (30%)**

Code base should implement all of the above (both pass and merit), as well as demonstrating at another **four of the concepts from categories A-E outlined below.**

**Concepts (Categories A-E)**

1. **Full-stack development:**

* DOM manipulation
* Hand-written CSS to style the Dashboard
* A CSS framework to style the Dashboard
* Refactoring the provided code base
* New tools, frameworks or approaches rather than using the given code base
* Continuous integration / continuous deployment

1. **Test-driven development**

* Unit testing
* Integration testing

1. **Programming paradigms**

* Object-oriented programming concepts including:
  + Inheritance
  + Polymorphism
  + Abstraction
  + Interfaces
* Functional programming concepts e.g. map, reduce
* Functional programming techniques in the context of a JS framework e.g. React Hooks

1. **Advanced JavaScript features**

* Higher order components
* Closures
* Async/await or Promises
* Types using TypeScript
* Advanced TypeScript features such as enums
* Animating objects in P5
* Manipulation of complex data structures
* Event handling
* Routing

1. **3rd Party API Integration**

* localStorage to persist state between sessions
* a database to persist data between devices
* an authentication service such as Auth0 or Firebase Auth
* the Open Weather API
* any other API

**Hand in: Parts A and B**

Please submit one PDF file for each part to the Google Classroom.

**Hand in: Part C**

Please submit your code via the provided Github Classroom repository. You should also provide a short time sheet documenting what work you did, and your git commit history should be clear and descriptive.