INB/N270 Class Assignment – PigWorld

Due Date: 29th October, 2013 – No late submissions accepted.

Weighting: 40%

Specification Version 1.0 (15th September 2013)

*In this assignment you have the option of working in pairs. If you work alone, you will not receive any special consideration if you are unable to complete the assignment in time. Programming pairs need to be registered with Mike Roggenkamp by email before COB 18th October 2013.*

*Given that you have 5 weeks to complete this assignment, there will be no extensions granted under any circumstances. If illness or other exceptional circumstances prevent you from submitting this assignment on time, email Mike,* [*m.roggenkamp@qut.edu.au*](mailto:m.roggenkamp@qut.edu.au)*.*

*Plan to submit by the end of Week 13: 25th October 2013.*

**Table of Contents**

1. Introduction 1

2. The Task 1

3. An Overview of PigWorld and its GUI Features 2

4. The Previous Programmer’s Notes 6

5. Finer Details About the GUI 6

6. Where to Start 7

7. Notes 7

8. Submission Details 8

# Introduction

This assignment aims to give you a “real world experience” that occurs far too often in the workplace. You have been hired to complete as much as possible of a project that has not been fully specified at this stage. You have been given (in your mind) an impossible delivery date. There is some supporting documentation to the project in addition to this document. In order to fulfil your contract you must deliver the prototype project which fulfils the stated functionality by the required date.

By the way, this is called the **Class Assignment** because it shows the importance of *using multiple C# classes* (contrasted to your first assignment, where all your code was in a single class).

# The Task

The Good Product Software Company has hired you to complete the implementation of a prototype that is at an early stage of development. It is a simulation, named **PigWorld**. The programmer who was to implement the prototype has left the company at short notice and has left behind only the source code of various C# classes and some notes regarding the prototype.

You are to continue developing the **Windows Form** version of PigWorld. You are required to:

1. Use **Windows Forms**, rather than any other GUI technologies such as WPF, XNA, web-pages, game engines, etc. You are expected to work with the existing C# classes, rather than creating new ones.
2. Develop a GUI layout that is very similar to the screenshot shown in **Figure 1**.

# An Overview of PigWorld and its GUI Features

This section provides an overview of how the final version of PigWorld should work, from a user’s perspective. To avoid this section being too long, some aspects – such as how pigs have babies – are described in later sections.

The prototype code, left by the departing programmer, does not implement all of the functionality described here. I.e. you’ll find some important bits missing in that code, which you’ll need to develop.

### 

Figure 1: A typical PigWorld scenario

*PigWorld’s Cells/Squares*

**Figure 1** shows a screenshot of a typical PigWorld scenario. In this scenario, there is one boy pig and one girl pig. In a black and white hardcopy of this document, the difference between the boy and girl pigs may not be obvious, but on a colour screen the colour band on the bottom of a boy pig is blue. For a girl pig it is pink.

There are two pig food trees, and beside one of the trees is a pill-like object, indicating that one of the trees has dropped some pig food. In PigWorld, hungry pigs move (one square/cell at a time) toward the nearest pig food. If a pig gets to the food before other pigs, he/she eats it. When eating, the pig does not move (for one step of the simulation).

Food contains energy. Each time a pig eats food his/her energy-level increases by the amount of energy in the food. Moving around PigWorld takes energy, so when a pig is not eating food, their energy-level slowly drops. The number on the right of a pig icon shows the energy-level of a pig. In **Figure 1**, both pigs have zero energy.

There is a fox in the lower right-hand corner of the world. For simplicity, foxes do not have genders, but they have some things in common with pigs, such as having an energy-level.

Pigs have poor eyesight. They can’t directly see other objects – such as other pigs, pig-food, foxes, or walls – until they are right-next to those things. Instead they use radars, or other ways, to head towards the things they are interested in.

The dark rectangles are walls. Animals (pigs and foxes) can’t move through walls, but they can use their radars to detect objects even though there may be one or more walls between the animal and the thing they’re interested in. I.e. radars can see through walls. This means that a radar gives the animal some guidance as to which direction it should move, but the radar doesn’t give complete directions to the animal. PigWorld would be less realistic if the animals had perfect knowledge of what to do next. (More details of how radars work are given later.)

The same poor eyesight applies to foxes. (So foxes also have radars.) When there is a wall between a fox and a pig (as in **Figure 1**), the fox won’t necessarily move *around* the wall to get the pig. But when a pig moves into a more favorable position for the fox, then the fox will move.

As pigs move around the world, they trail a rope behind them. The rope serves as a reminder to the pig of squares that it has already visited. Pigs have no other direct memory of where they have been. In **Figure 1**, both pigs are trailing some rope, across 2 squares each. Boy pigs trail blue ropes and girl pigs trail pink ropes. A rope belongs to exactly one pig. Each pig recognizes its own rope as different from other ropes.

*Context Menus*

Each of the life-forms has a (right-click) context menu, shown in **Figure 2**. The same context-menu applies to both girl and boy pigs.

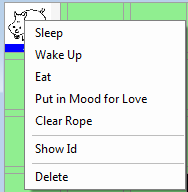
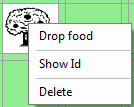
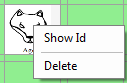
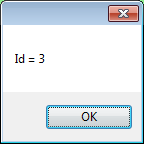
   

Figure 2: The three context menus and the Show Id message box

Each **Show Id** menu item displays the life-form’s internal Id. This may be useful for debugging purposes, as described elsewhere. Each **Delete** menu item removes the object from PigWorld. The meanings of the other menu items should be fairly clear from their names.

*PigWorld’s Controls*

At the bottom of **Figure 1**, there are various radio-buttons. These buttons allow you to add objects to PigWorld, at any time (even while the simulation is running). To add an object, click on the appropriate radio-button, then click on the square/cell where you would like to place the object.

In the special case of adding a wall, click on the rectangular space between any two squares. Similarly, clicking on an existing wall will remove it. Walls can be added or removed at any time (even while the simulation is running). By the way, clicking in the tiny squares at the corners of each cell does nothing.

The controls in the right-hand panel work as follows:

1. The **SetupDemo** button and its associated **UpDownControl** initialise PigWorld to one of a number of given scenarios. Any objects and walls that are already present when this button is clicked are removed first.

This button is very useful as a timesaver. It allows you to have standard setups which are created much more quickly than having you use the radio-buttons. This is especially useful when you are developing your program and restarting it frequently to test it out.

1. The **Start** button starts a timer that runs continuously. By default, the timer ticks once a second (i.e. once every 1000 milliseconds). When the timer ticks, it is the same as if the user had clicked the **Step** button.
2. The **Stop** button stops the timer.
3. The **Step** button advances the simulation by one step. E.g. the pigs move by a single square, or possibly eat some food. Things can happen too quickly for you to see what is really going on. The **Step** button allows you to control the speed at which things happen. You can hit **Start** at anytime, if you grow tired of **Step**. Or you can start a simulation running by hitting **Start**, then hit **Stop**, and continue with **Step**. (More details on how this is implemented inside the program code are provided separately – see section 4.)
4. The **Remove Walls** button deletes all the walls, but leaves any other objects where they are.
5. The **Remove All** button removes everything, including the walls.
6. The **Quit** button closes the form, which causes the program to exit.
7. The **Faster/Slower** TrackBar determines the speed of the simulation, i.e. controls the interval at which the timer fires. This works the same as the TrackBar that you created in the Week 9 prac.
8. The **Enable Read Audio** checkbox controls whether audio files are played at certain key moments during the simulation. Hearing that audio can enhance the simulation, but not if you are using your computer’s speakers while sharing a room with other people. Hence, real audio is an option for the user. (There is no other affect on the simulation.)
9. The **Show Debug Info** checkbox controls whether additional information is displayed on the screen. That information is intended to provide a small amount of assistance when you are developing your program. In a real-world application, this checkbox would be removed before the program was released to end-users.

**Figure 3** shows the same screenshot as in **Figure 1**, but with this checkbox enabled. Each empty cell shows its **[column, row]** number at the top, e.g. cell [2, 3] is row 2, column 3 (counting from zero, in each case). The number **0** at the bottom of each cell indicates that the sound-level is zero in each cell at the moment. (Sound-levels are described later.)

Although hard to in **Figure 3**, the extract in **Figure 4** shows that additional text is being displayed for each of the pigs, showing what they are doing, currently.

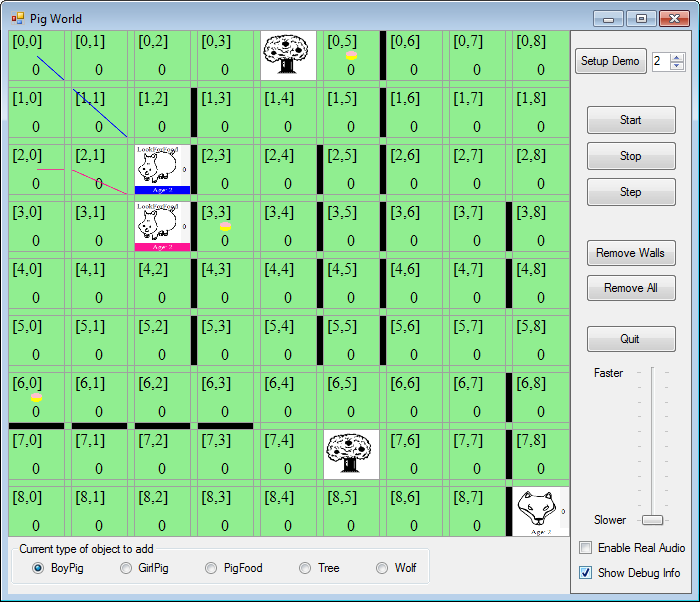


Figure 3: PigWorld with Show Debug Info enabled



Figure 4: Zoomed-in part of Figure 3, showing that both pigs are looking for food, currently

# The Previous Programmer’s Notes

The following text is the introduction to the previous programmer’s notes about the prototype and its classes. See the document, **My Working Notes,** for the full set of notes.

*In the* ***Solution Explorer*** *window, the current Visual Studio solution contains two “projects” (or “folders”, if you like):*

* *The* ***PigWorld*** *project contains the (incomplete) code that performs the logical processing of the simulation. The classes in this project know nothing about the GUI and you must keep them that way. In particular, these classes* ***do not*** *use any WinForms objects (i.e. do not reference the namespace* System.Windows.Forms*) and you must not change that. Similarly, these classes* ***do not*** *refer to any classes from the GUI project described below and you must not change that.*
* *The* ***PigWorldGui*** *project contains the (incomplete) code for displaying the simulation as a GUI. These classes* ***can use*** *any WinForms objects (from the namespace* System.Windows.Forms*). Similarly, these classes* ***can refer*** *to any classes in the PigWorld project described above.*

# Finer Details About the GUI

The screenshot in **Figure 1** illustrates the basic layout of the GUI, there are some finer details that the screenshot does not show, or does not show clearly. Do not start implementing the GUI until you have finished reading this document and the programmer’s notes concerning the ***PigWorldForm class*** in **My Working Notes.**

1. Position the GUI Controls as shown in **Figure 1**. While you are not expected to get their locations and sizes correct to the very last pixel, your GUI layout should look fairly close to the one given in this specification. Apart from the message box shown in **Figure 2**, no additional dialog boxes or message boxes are to be used (unless you need them to assist you with debugging, although they are not to appear in your submitted assignment when it is executed).
2. The PigWorld cells/squares are displayed by using a **TableLayoutPanel**, but we need a fairly special **TableLayoutPanel**. So don’t waste your time by adding a **TableLayoutPanel** in **Design View**, unlike the way you did it in some of the Pracs. Similarly, don’t waste your time by adding a control for each of the 49 (= 7 x 7) squares on the board, or the walls, in **Design View**. It is much easier to do this in your program code later. This is explained in the programmer’s notes.

# Where to Start

Use **Design View** to add controls to the prototype **PigWorldForm**, *so that it has the overall layout as shown in* ***Figure 1****.* Don’t expect to get your form looking exactly the same as the screenshots, to start with. The PigWorld cells/squares *will not appear* in **Design View**.

As well as reading what the previous programmer said about this form, you should refer to Lectures 8 & 9 and the related Practicals, for help on using **Design View**.

As stated in section 5(b) above, do not waste your time by trying to create the PigWorld cells/squares in **Design View**.

Once you’ve got all (or almost all) the controls added to the form, it’s time to start adding some code to make those controls work a bit better. Most of this work will require you to add event-handlers to those already provided in **PigWorldForm.cs**, but a lot of the functionality will be implemented in other (existing) classes. You probably won’t understand all the existing code, so make use of the previous programmer’s notes for assistance. There are also comments in the code that should help explain the purpose of the methods provided.

The challenge in this assignment is to complete what you can before the due date. Better to hand in a working project which does some things correctly than one which does not run.

# Notes

Though all care has been taken in the production of this specification, there may be a need to notify by class email any alterations/clarifications to this specification.

**SO CHECK YOUR EMAIL DAILY.**

**Working in Optional Pairs**

Ensure that both people in the group are involved and are responsible for doing some part of the assignment. Do not forget to register your group details via email to Mike before COB on 18th October 2013. Include the full name and student number of each person in your email.

# Submission Details

Will be provided by 15th October 2013 on Blackboard.