

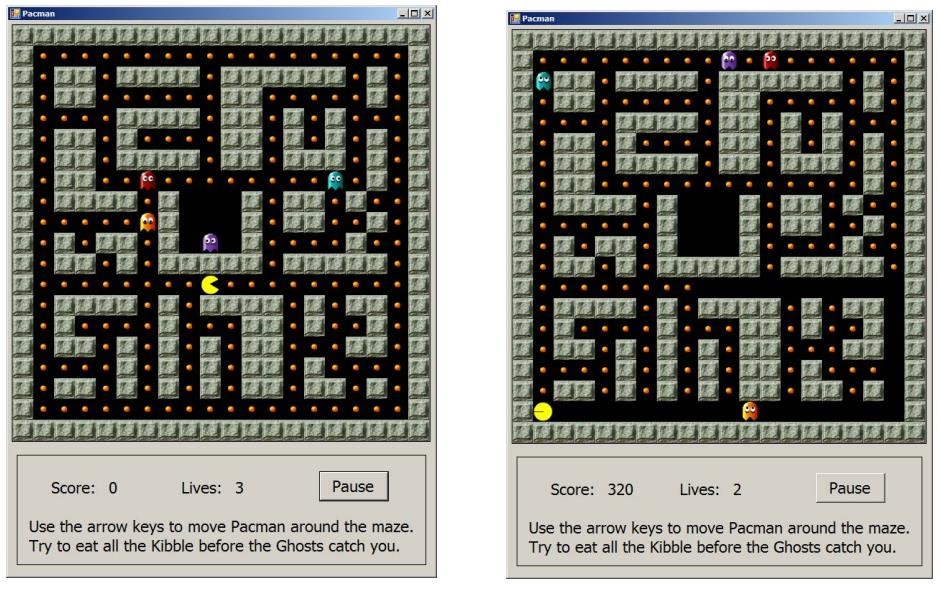
**IA511001 Programming 2**

**Assignment 2 –** **Pacman Game**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Study block** | Study Block 4 - 2024 | | | |
| **Date issued** | Monday 15th April 2024 | | | |
| **Due date** | Wednesday 27th November 2024 and Wednesday 4th December 2024 | **Time** | | 11:59 pm (midnight) |
| **Delivery:** | Submit to Moodle before deadline | | | |
| **Weighting** | 35% | | | |
| **Marks out of** | 100 | | | |
| **Instructions** | Complete this cover sheet and submit with your assignment.   * This assignment must be your own work. * Collusion, copying or plagiarism may result in disciplinary action. * We advise that you keep a copy of this assignment. * Refer to following website for reference related resources:   <http://www.cite.auckland.ac.nz/index.php?p=quickcite> | | | |
| **Lecturer** | Dr. Barry Dowdeswell | **Class time** | Wednesday 1:00 pm to 5:00 pm  Friday 1:00 pm to 5:00 pm | |
| **Student declaration:** | By entering my name in the box below I confirm that:  *This is an original assessment and is entirely my own work.*  *Where ideas, tables, diagrams etc. of other writers have been used, I have acknowledged the source in every case.*  *This assignment has not been, nor will be, submitted as assessed work for any other academic course.* | | | |
| **Student ID No** | **1000124188** | | | |
| **Student Name** | **PRABHJOT SINGH** | | | |

# Assignment Brief

For this assignment, you will recreate the classic arcade game Pacman. In this game, a little smiley- face (Pacman) travels through a maze eating kibble. Also in the maze are three ghouls. Pacman’s goal is to consume all the kibble in the maze without running into any of the ghouls. The screen shots below show an example of the simplified version of Pacman required for this assignment. The image on the left shows the start of the game; the image on the right shows the game after Pacman has inadvertently run into the light-coloured ghoul.



# Learning Outcomes

At the successful completion of this course, students will be able to:

1. Build an interactive, event-driven GUI application using pre-built components.
2. Declare and implement user-defined classes using encapsulation and inheritance.

Instructions

1. You can work in a group of **two** on this assignment or do it individually. If you work in a team of two, then you must each submit a brief, individual reflection report showing what parts you worked on. The template is available on Moodle.

Details

Pacman is controlled by the arrow keys. In response to an arrow key, Pacman moves one step in the corresponding direction. If he moves over a piece of kibble, the kibble is consumed (i.e. it disappears), and Pacman’s score is increased by one. Pacman cannot move through walls (the solid, green-colored areas in the screen shots above). If the pressed arrow key directs Pacman into a wall, Pacman simply does not move.

Pacman must be animated (his mouth must open and close as it moves).

# Behavior of the Ghouls

In the basic implementation of our version of Pacman, the ghouls simply move randomly, one step at each timer tick. Like Pacman, the ghouls cannot walk through walls. The game is much more interesting to play if the Ghouls have some rudimentary artificial intelligence. See “Extra Credit Functionality” below for suggestions.

# Functional Requirements

1. Your solution must be Object-Oriented.
2. Your game must provide a 20 by 20 grid of square areas that comprise the maze. Each grid square will, at the start of the game, be either a kibble, or a wall. See below for more details.
3. Your game should be driven by a single Timer.
4. Your game must provide a Pacman who behaves as described above. You should declare a global variable of appropriate type to implement the Pacman.
5. Your game must implement three Ghouls who move as described above. You should manage your Ghouls within an array or a list.
6. Your game must monitor for arrow key presses to direct Pacman.
7. Pacman must have an animation involving at least two different images. (In the demo on the Moodle Resources Tab, Pacman has an Open Mouth and a Closed Mouth image.) Implement the animation by alternating the images that Pacman draws for himself at each Timer tick.
8. Each time Pacman walks onto a square containing kibble, the kibble is consumed, and Pacman’s score increases by 1. Pacman’s score must be clearly displayed and labelled.
9. Your game must determine win/loss as follows:
   1. The game is won if Pacman consumes all the kibble.
   2. The game is lost if Pacman runs into (i.e. is on the same grid square as) any Ghoul before consuming all the kibble.
10. Your game must give appropriate win/loss feedback.
11. Your game must stop, when either the win or loss conditions are met.

# Class Structure

Your program must be Object-Oriented. You must, therefore, declare the necessary Classes and instances of these Classes. You will need, at a minimum, objects to represent Pacman, the Ghouls and the Maze. You should also declare a base class Creature. Pacman and Ghoul must inherit common properties and capabilities from the abstract Creature class. Note that marks will be allocated for the Object-Oriented correctness of your implementation. See the example Inheritance code provided to you in a previous session.

Your Classes will need Fields, Properties and Methods. For example, both Pacman and Ghoul objects must know where they are in the maze, how to move themselves, how to draw themselves, and so on. They will need to know how to determine if they have collided with another game entity. Pacman will need to know his score, and how to correctly implement his animation.

You need to document examples of these fields, properties, and methods in the pseudocode of your application. Provide this in your Design Document.

# Implementing the Maze

An easy way to implement the maze is to use a simplified version of a common computer graphics technique called Tile Mapping. In this technique, the area to be drawn is divided into small squares, and an image is assigned to each square. The drawing is accomplished by displaying each image at the correct grid location.

Suppose this image will be used for the wall  and this for the kibble .

If you define a 3 x 3 grid with the top row as wall, the middle row as kibble and the bottom row as wall, it would appear when drawn as:



By making a larger grid, you can build a maze.

The grid in this version of the game is 20 by 20. Images you can use for the grid, for Pacman, and for the Ghouls are on the Moodle (Resources Tab) Alternatively, you may source your own images.

Your grid should be an object; The class should hold the current state of each grid location. It must also know how to draw itself to the screen.

# Getting Input from the Keyboard

When a user presses a key on the keyboard, a KeyDown event is generated. For the Form’s KeyDown event, the event method signature is:

private void Form1\_KeyDown(object sender, KeyEventArgs e)

You need to write a handler for this Form event. The argument you are interested in is KeyEventArgs e, which contains the value of the pressed key. The arrow key values are Keys.Left, Keys.Right, Keys.Up and Keys.Down for the corresponding arrow key. When the user presses the Down arrow key, for example, the system generates a Form1\_KeyDown event, and passes in the argument e, whose value is Keys.Down. In your Form1\_KeyDown event handler, you can have statements like:

switch (e.KeyCode) {

case Keys.Left:

*// do something in response to the left arrow key* break;

……..

The Form’s KeyPreview property must be set to True. Otherwise, it won’t be able to respond to the KeyDown event.

# Optional Extra Credit Functionality

If the ghouls simply move randomly, they won’t make it very challenging for PacMan to eat all the kibble. It will be a much more interesting game if you program the Ghouls to chase PacMan around a little. This is easy to do. Here is a pseudo-code description of the common algorithm for chasing:

At each turn:

* Decide if Ghoul should chase along the up/down axis or the left/right axis  If you decide up/down:
  + Find out where Pacman is (he should provide a method for this).
  + If PacMan is above you, move up
  + If PacMan is below you, move down
  + Remember you can’t move through walls…
* If you decide left/right
  + Same logic as up/down, except on the horizontal axis
* You will find that if you implement a Chasing algorithm like this, the three Ghouls will rapidly corner PacMan, and the game will be essentially impossible to win. Therefore, you will need to adjust your algorithm as follows:

At each turn:

* Decide whether to chase or to move randomly
* If chase
  + Chase, as above
* Otherwise
  + Move randomly.

* You should make the decision whether to chase probabilistically. That is, you might want to chase 10% of the time, or 50% of the time, or whatever gives the game a level of difficulty you like. To make a probabilistic decision do something like this:



//

this

will

be

true

/MYPROB

1

of

the

time



//

or

2

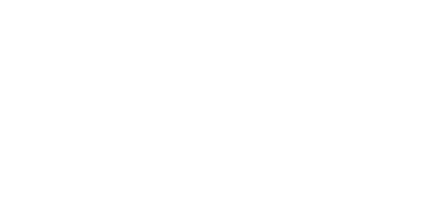
or

,

50

or

whatever.



const

int MYPROB =

;

10

…..

DecisionTemp = r.Next(MYPROB);

if (DecisionTemp == 0) then

You can find Pacman online at:

<https://www.webpacman.com/pacman-html5.php>

# Submission

1. Submit your design documentation with UML class diagrams and pseudo-code showing the complete design of your application on Week 8 Wednesday 11th September before 5:00 pm.

1. For your project you must submit your application through the Moodle submission link. Compress the Visual Studio Project folder containing all the project files.
2. If you work in a team of two then you must **each** submit a brief, individual Reflection Report showing what parts you worked on. The template is available on Moodle.

Marking Schedule Assignment 2:

|  |  |  |
| --- | --- | --- |
| **Feature** | **Comment** | **Percentage of Total Mark** |
| Pseudocode | Deconstruction of the problem  Abstraction of details  UML class diagrams  Iterative refinement of methods  Design of algorithms (eg collision detection) | 10 |
| Code commenting | Block comments at beginning of program.  Block comments for every class.  Block comments for every method.  In-line comments in important code functions. | 5 |
| Functional  Requirements | 20x20 or larger maze  Pacman walks on keypress, is animated Pacman eats, and can be eaten by a Ghoul  Ghouls move on timer  Scoring system  Appropriate feedback  Game stops | 40 |
| Object-Oriented  Correctness and Code  Elegance | Class design – Pacman and Ghoul inherit common characteristics from the abstract Creature class.  Modularity  Constants  Enumerations  Algorithmic  Elegance  Error handling | 30 |
| Product Aesthetics | Attractive screen layout and colour scheme.  The user should immediately see how to play. Instructions should be provided if you think they are needed to explain special features.  Any additional instructions are included.  Clear, appropriate feedback to user. | 5 |
| Additional  Functionality | Ghouls chase  Pacman lives Levels  Something else that makes your game different  ……. | 10 |
|  | **Total Mark** | 100 |

**Assignment 2 Assignment Submission Evidence:**

*Paste screen shots on this page of your application when it is running. Use more pages if you need to.*

*Include example source code that shows your comments and examples of methods and calculations.*



A screen shot of a game

Description automatically generated

