Core Spikes

Introduction

This document is an overview of the Spikes available in this subject.

What is a Spike?

A Spike is a small project designed to close, through research & implementation, a knowledge or skill gap. The emphasis here is on the small – you should only do what you need to close (and demonstrate that you have closed) the skill gap.

Each Spike consists of a Spike Report and a deliverable. The deliverable is usually a code piece, but can be a research report (or, occasionally, both). If the deliverable is a research report, the research report is a separate document to the Spike Report – you must submit both reports.

The Spike Report is a report that details the work you have done to complete the spike. A template for this report is available on blackboard.

All of the spikes below are compulsory. Completing them is required to get a Pass. Doing these spikes to a high degree of quality and doing additional spikes from the Non-Cores Spike list (see Blackboard) is required to get a Credit. Distinction and High Distinction require a Credit level of work plus a project and a research project, respectively.

These spikes broken up into Blocks to indicate Spikes that have common themes. All Blocks and Spikes in this document must be completed to get a passing grade,

How to do C++ work & Game Loops

Description:
This block of spikes is designed to familiarise you with the programing language and IDE you will be using for the rest of the semester.

Spikes:
*Spike 1: Simple Game Loop
*Spike 2: Intro to Visual Studio
*Spike 3: Debugger Use
*Spike 4: Non-Blocking Game Loops

Spike 1: Simple Game Loop

CORE SPIKE

Context: Games are commonly driven by some form of game loop.

Knowledge/Skill Gap: The developer is not familiar with basic game loops.

Goals/Deliverables:
You need to create the "GridWorld" game, as described in a simple specification document available **below**. The game will demonstrate the use of a simple game loop, separation of update/render code, and game data (both for the world map and players current location).

You will need to deliver the following items:

1. A simple paper-based plan for your code design. (Yes, a simple functional design is fine – just as long as you can demonstrate that you planned first before coding!)
2. Create a simple console program that implements the "GridWorld" game using a simple game loop. The game must demonstrate the separation of:
    a. processing of input (text commands from the player),
    b. updating of a game model (where the player is and their options),
    c. display (output current location and options) of the game to the user.
3. Spike Outcome Report.

Note: The Spike Outcome Report is always a required "deliverable". It will not be repeated in future spike requirements as it is assumed.

Recommendations:

- Read the "GridWorld" game details and on paper (as required in the deliverables) do a quick sketch/design of how you will organise your code. (No formal standard specified… just use something that would work to help you explain your design to another programmer)
- Look at the Spike Outcome Report template – note what you need to record for later.
- Use an IDE for C/C++ development that you already know. (We'll look at Visual Studio later so don't get distracted from the main point of this spike!)
- Begin small, test often. Use simple printouts to check values (or the debugger if you already know that and are comfortable). Don't try fancy debugging yet if you don't know it – that's a later spike if it's something you need to work out. Stay on target!

Tips:

- Consider a DEBUG macro condition if extra code is needed for testing purposes.
- You do not need to show the "map" or the current player location – but it is useful.
- If you find some useful resources that helped you to get your code working, then remember to note them down and include them in your spike outcome report. Google, books, blogs, classmates, etc. all go in the spike report.
- Have a plan for when you present your outcomes to the tutor. (They will probably say something like "Right – show me"… and it's up to you to show-off what you've done.)
- Many of these tips also apply to spikes in general and are not exclusive to this spike.

DO NOT

- Make things any more complicated than you need – just get it working.
- Create complex data-structures. A simple 2D array is fine (and expected).
- Add all the fancy features you can think of! Save ideas for later and perhaps develop them as a portfolio item. (You should still document/note them down though.)
- Load the map data from a file. Simply hard code the map data.

Spike ILOs

Design:	LOW	(1)
Implementation:	MEDIUM	(2)
Maintenance:	–	(0)
Performance:	–	(0)

Extensions:

Rewrite (or refactor, if possible) your GridWorld as an Object Oriented version. Focus on the design of your new code, rather than the complexity (i.e. don't over-engineer). Therefore keep in mind good design principles for your code:

- Robust
- Maintainable
- Scalable
- Readable

Hint: A quick paper-based plan will benefit you and impress your tutors!

Spike ILOs

Design:	LOW	(1)
Implementation:	MEDIUM	(2)
Maintenance:	LOW	(1)
Performance:	–	(0)

Spike 2: Intro to Visual Studio

CORE SPIKE

Context: There are many IDE tools available and developers need to be able to make informed decisions in order to be productive with the tools they select.

Knowledge/Skill Gap: The developer is not familiar with Visual Studio 2015.

Goals/Deliverables: [SPIKE REPORT] + [COMPARISON REPORT]

1. Using the simple command line program from Spike 1 create a working project using Visual Studio (2015)
2. Create a short report to document your findings. (Note: the short report is NOT the spike report - you should have two reports for this spike.) Include the following details:
    a. A point-form list of the steps required to create a new project in Visual Studio, add new files, compile and run a command line program.
    b. A point-form list of the steps required to create a break-point location in the program, and run the program in using Visual Studio's Integrated debugging system so that program execution stops at the nominated point.
    c. Note how to inspect the value of variables during debugging.

Recommendations:

- Assume you are writing steps to help a colleague to create a project.
- As a review, show your steps with some other students. Make sure your notes make enough sense! Update any suggestions or omissions so that your notes are valuable.
- Be sure to note the version numbers of software you use in your spike outcome report.
- Remember that spikes should be about the doing the minimum required to close the gap in knowledge or skill (or technology). However if you do not write enough so that someone else would understand, you have not complete to spike work correctly.

Spike ILOs

Design:	–	(0)
Implementation:	LOW	(1)
Maintenance:	LOW	(1)
Performance:	–	(0)

Extensions:

Investigate and write a point-form list of the steps required to use at **least one** "deeper" feature of Visual Studio 2015. For example:

- NuGet Package Manager
- Source Control Integration
- "Advanced" debugging
- How to add additional include directories for a project
- Adding your own pre-processor macros depending on build configuration
- Multiple configurations (Development vs Release) and how they can be used

- Etc.

Note: It may be necessary to **fully** (within reason) explore these features depending on how complex/advanced they are. You could also write steps for **two** or more features.

Spike ILOs

Design:	–	(0)
Implementation:	MEDIUM	(2)
Maintenance:	MEDIUM	(2)
Performance:	–	(0)

Spike 3: Debugger Use

CORE SPIKE

Context: Effective use of the debugger is essential for isolating and repairing code errors within a non-trivial project of source code.

Knowledge/Skill Gap: The developer is not familiar with the use of debugging tools.

Goals/Deliverables: [FIXED CODE] + [SPIKE REPORT]
You need to demonstrate that you have done the following:

1. Download, compile and run the "spike3" program provided on blackboard in Visual Studio. The program contains a number of "bugs", including a deliberate memory leak. (Memory is allocated, but not de-allocated.) You must discover and fix errors. (You might not find them all!)
2. Use IDE debugging tools to identify memory leaks and other issues. You must document in your spike report the IDE steps you used to do identify bugs. (You are welcome to use screen shots to supplement the written steps you have used. See the planning notes for suggested steps.)
3. Save the fixed code and included source code comments to document what you changed.

Recommendations:

1. Open the program in Visual Studio.
    a. Add a break point at a point in the source code at a point likely to be the cause of the leak. (If you're unsure, place one close to the start of the application and step through all of it. This does mean more stepping, but at least you are less likely to skip over the broken code!)
    b. Compile and run the program in debug mode
    c. When the program breaks, step through the program, examining the source code and variables for potential causes of the leak. Note them down.
    d. Repair any leaks found. Add comments about your victory!

Note:

- You might not understand all the code provided, but that's okay! You should still be able to step through the program execution to find some bugs, if not all.
- The goal of this spike is not really about fixing the bugs/leak(s), but rather to get you familiar with the process of using the debugger, and to demonstrate its usefulness. Your spike report should not dwell on the nature of the bugs (or your corrections), but rather show that you can effectively use the debugger to solve problems (the spike "gap").

Spike ILOs

Design:	–	(0)
Implementation:	LOW	(1)
Maintenance:	LOW	(1)
Performance:	–	(0)

Extensions:
Find and fix all bugs present.

Spike ILOs

Design:	–	(0)
Implementation:	LOW	(1)
Maintenance:	LOW	(1)
Performance:	–	(0)

Spike 4: Non-Blocking Game Loops

CORE SPIKE

Context: The non-blocking game loop is a more sophisticated implementation of the game loop concept. It is the most common form of game loop used by modern games.

Knowledge/Skill Gap: The developer is unfamiliar with the non-blocking game loop.

Goals/Deliverables:
[CODE] + [SPIKE REPORT]

1. Create a console program that implements the "Gridworld" game using a non-blocking game loop. The loop must execute continuously, only processing input when it occurs, and only providing output when necessary. The Gridworld game should be implemented with a timer.

Recommendations:

- The input block in C++ makes it easy to determine whether input has occurred and act on it if it has - you'll have to go beyond simply using the >> operator, though.

Spike ILOs

Design:	–	(0)
Implementation:	MEDIUM	(2)
Maintenance:	LOW	(1)
Performance:	–	(0)

Extensions:
Complete this spike using a "**non-hacky**" method e.g. Multi-threading, Framework/library, OS specific API

Spike ILOs

Design:	–	(0)
Implementation:	MEDIUM	(2)
Maintenance:	HIGH	(3)
Performance:	–	(0)