Assignment Submission

Student Names:	Kieran McEvoy Tríona Barrow
Student Numbers:	12362641 11319851
Degree Programme:	B. Sc. in Computer Applications (Software Engineering)
Project Title:	Third Year Project
Module Code:	CA326
Lecturer:	Charles Daly
Project Due Date:	16th November 2015
Group Number:	46
Assignment:	Functional Specification

Declaration

We declare that this material, which we now submit for assessment, is entirely our own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text. We understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should we engage in plagiarism, collusion or copying. We have read and understood the Assignment Regulations set out in the module documentation. This assignment, or any part of it, has not been previously submitted by us or any other person for assessment on this or any other course of study.

1. Introduction

a. Purpose

The following functional specification is designed to give a broad overview of “All Alone?” with regards to the system requirements and its analysis. Its purpose is to serve as a reference for the system design, including the core functions that it will need to implement. Where possible, high-level descriptions of how these functions are expected to be implemented will be provided. The intended audience for this document are the system designers, project coordinator, project supervisor, and the CA326 module demonstration panel.

b. Overview

‘All Alone?’ will be a side-scrolling exploration game, with emphasis on puzzle solving and getting through obstacles in the player’s way. It will be coded in C# using the Unity 5 engine. It will target the gaming market, which although is extremely competitive and crowded, is a vibrant and growing environment to work in.

The main goal of the project is to construct and develop a game which will match the quality of existing games - both with regards to 2D style games and puzzle games that have already been released. As well as this, C# is an increasingly popular language for development, even outside of the Unity platform. The combination of the this status, the increasing number of games developed in the Unity engine, and the popularity of games themselves will lend towards a successful project.

c. Business Context

This game has a large potential for success. The indie gaming industry is an ever growing and thriving industry, with several success stories. There are multiple new Kickstarter projects, start-ups and side projects everyday related to gaming in all forms. Although this makes it a clustered and highly competitive market, success is a common occurrence. Steam Greenlight and Kickstarter are just two examples of ways an indie developer can promote and sell his or her project.

d. Glossary

Side-Scrolling - Think of the Mario games, where the game runs in a 2D environment, and the camera follows the player. Left and right are the only direction a player may move, apart from jumping up or dropping down.

Steam Greenlight/Kickstarter - Online platforms where developers can submit their projects and allow consumers to support it with their own money or votes. It has simplified the process between having a game and selling it to the market.

Player - Following conventions of the gaming industry, users of the game will be referred to as players.

2. Description

a. Product/System Functions

The game will focus around a 2D sprite that is controlled by the player as they navigate through the game. The levels will be fixed in order and generated during development, creating a linear progression through the game. As part of the levels, intellectual style puzzles will have to be completed to allow the player to progress. As part of this, a number of functions will have to come into play to allow this, as follows:

- General
    - Run Game
    - Close Game
- In-Menu
    - Start Game
    - Load Game
    - Adjust Volume
    - Close Game
- In-Game
    - Move - Left/right and jumping up
    - Interact - Moving items/buttons in the level
    - Player Death
    - Restart Level
    - Restart Game
    - Save Game

b. User Characteristics and Objectives

It's foreseen that the main user for the game will be those who already have an interest in gaming in some shape or form, whether specifically puzzle games or otherwise. Generally our target audience would be young adults of either gender - so from 15 - 30 years of age. Smooth graphical performance and high quality puzzles would be expected to make this a compelling game to play within this target market.

The target market will likely have experience with games previously, so would be familiar with the application domain and the general software system involved.

c. Operational Scenarios

- Successful Playthrough

The player has proceeded through all available levels and has reached the surface, having progressed up and out of the underground caves. The player will be able to move around on the surface for about 30 seconds after this, at which point they will be presented with a “Game Completed” message and the following options:

  - Restart Game - start the game over from scratch.
  - Quit Game - go back to the starting screen for the game.

- Player Dies

The player is making their way through the game and has reached the third cave section. To proceed through this cave, they must solve the problem within this cave, which involves interacting with switches on the cave wall in order to move objects within the level. However, the player pushes the switch furthest to the right and a boulder falls onto them, causing their sprite to die. The player will then be presented with four options:

  - Quit the game
  - Restart the game
  - Restart the level (resets the level)
  - Save the game (saves level reached)

- Player Quits Game

During the playthrough of the game, at any time the player can load up the menu and select the option to “Quit the Game”. At that point, if the player has not saved the game, they will lose their progress through the levels since the last save (if any) and will be presented with the start screen once more. From there, they can:

  - Close the game
  - Start a new game
  - Load a pre-saved game
  - Change some game options.

d. Constraints

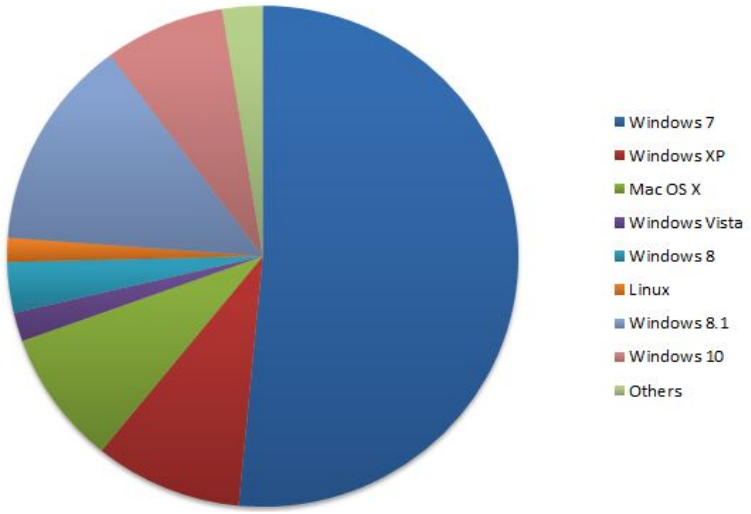
Time

With the impact of a full academic year and the related assignments, examinations and coursework, this may lead to timing issues - we will have to do the majority of the project alongside full time education. With the type of project, a game, it is possible that it may continue to evolve while in the development stage. The project will be completed by then, however it’s possible that this may impact on the time available to use for user testing before the submission in March.

Language
We will be coding the game itself using C# on the Unity platform - and while there are a lot of resources, both for the language by itself, as well as with Unity - we are still embarking on a project using a language we do not have experience with. However, from looking at C# in comparison to the other available languages (Javascript and Boo), we felt that this would be the best option for this project, and feel we can overcome any learning difficulties.

Quality
As mentioned by Brian Stone and Mark Roantree at our project presentation, with the time constraints in mind it will be difficult to create a high quality, polished game. We will hopefully be able to keep an eye on this and address this throughout the project, by keeping on top of our schedule and in contact with our project supervisor.

System Requirements
Unity itself has some basic requirements - both for development and for running the games. On the development side, we will both be developing on supported Windows platforms (Windows 10 personally and Windows 7 SP1 on the lab PCs) which should not cause any issues. Unity advises that the requirements really depend on the complexity of the project - as ours is quite simple we should be able to do this successfully. The requirements for running the game is similar, however we feel confident that the rate at which people are updating to more recent operating systems will remove this. (Shown below)



3. Functional Requirements

a. Main Menu

- **Description**
  This function is the first screen the user will see upon loading the game. It will have four options to choose from, “New Game”, “Load Game”, “Exit Game” and “Options”. The “New Game” option will start a new game of ‘All Alone?’. “Load Game” will load from a previous save. “Exit Game” will close down ‘All Alone?’ cleanly. “Options” will bring the user to the options menu, where they can adjust the volume.
- **Criticality**
  This part of the project is to ensure the user has the option to do what they need to before playing the game, such as adjusting the volume. Rather than throw the player straight into the game, we feel this little brief pause before the game is necessary.
- **Technical Issues**
  The main menu itself will require it’s own code and graphical user interface separate from the game itself. Otherwise, it will not be a difficult task to accomplish.
- **Dependencies**
  The New Game option will need to initiate the game itself.

b. Player Avatar

- **Description**
  This represents the protagonist of the game. It will be able to move left and right, and jump up into the air. It will be able to interact with the environment in a manner of ways, including but not limited to pulling levers, flipping switches and pushing objects.
- **Criticality**
  This is vital, as without the player avatar, the game would not function. The player must use the avatar to complete the main aspects of the game, puzzles.
- **Technical Issues**
  This will require us to program a fully functioning player avatar, that is capable of interacting with the world around it.
- **Dependencies**
  This function must be able to interact with the environment functions, physics functions and object functions, in order for the game to be playable.

c. Environment

- **Description**
  The environment function will serve as the platform that our protagonist will move along, and also the places where several objects will also move on or attach to. It is the area and setting of the game.
- **Criticality**
  This is important as almost every aspect of the game will be in some way attached to the environment of the area the protagonist finds themselves in. Without it, there would be no structure to the levels of the game.
- **Technical Issues**
  This function must work properly to ensure playability. So the objects of the game and player must be able to work with the environment efficiently, otherwise the player may fall where they shouldn't, or objects are non interactable to the player due to not working properly with the platforms it is attached to.
- **Dependencies**
  This function will depend less on other functions, as it will be the 'canvas' that other functions will be attached to or traverse on.

d. Objects

- **Description**
  This function is the representation of all the interactable objects that the player's avatar will be able to work with. These will include levers, rope, rocks and more.
- **Criticality**
  In order for the game to progress, the player will have to interact with these objects in order to solve puzzles and advance. They are an integral part of the game.
- **Technical Issues**
  This will probably be one of the most difficult aspects of the game, as there will be a large variety of in-game objects that will be used. They must be able to work with the environment, player avatar and any physics part of the game successfully, otherwise the game might not be able to progress at all.
- **Dependencies**
  The objects of the game, like the player avatar, will need to function with almost all the other objects in order to provide a smooth playing experience for the player him/herself.

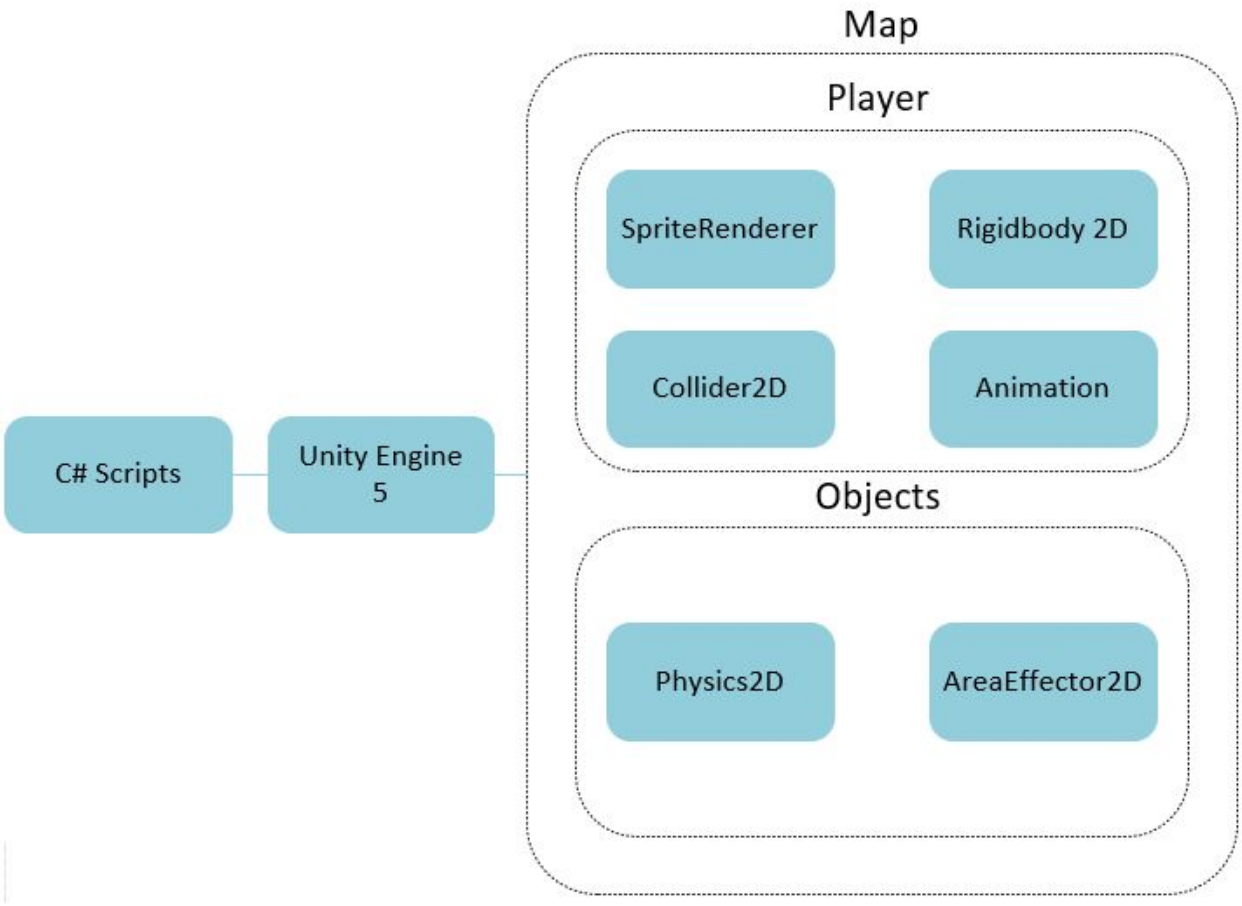
e. Physics

- **Description**
  Physics will be an important part of the project. Not only the player, but the in-game objects must all rely on physics in order to function. This will mostly be gravity, but will also involve some momentum.
- **Criticality**
  This is an integral function of the game, as in order for certain puzzles to work, physics will be involved. It will also affect how high the player will jump, and how fast objects and the player avatar will fall.
- **Technical Issues**
  There shouldn't be too many problems with the physics function. It will require some mathematical thinking and possibly some algorithms, but nothing of immediate difficulty.
- **Dependencies**
  Like the environment function, the physics function will have other functions relying on it more than it relying on them.

f. Camera Mapping

- **Description**
  This function will act as the player's point of view. It will only face one direction, emulating a 2D environment. It will move left and right, with the protagonist roughly in the centre of the screen. At certain points in the game, like when the player encounters a particularly large puzzle, the camera may zoom out, to allow the player a larger view of the area they find themselves.
- **Criticality**
  This is not the most important or difficult of functions to implement, as the camera will only have to follow the player avatar, and not do much else. The zoom feature might be helpful, but will not be completely necessary for the end product.
- **Technical Issues**
  This will not be a difficult task to implement into the game. Additional features like zooming or panning might be integrated into the game, but will add more work time and difficulty to the project at hand.
- **Dependencies**
  This function will depend entirely on the player avatar function, and its current location in the in-game area.

4. System Architecture

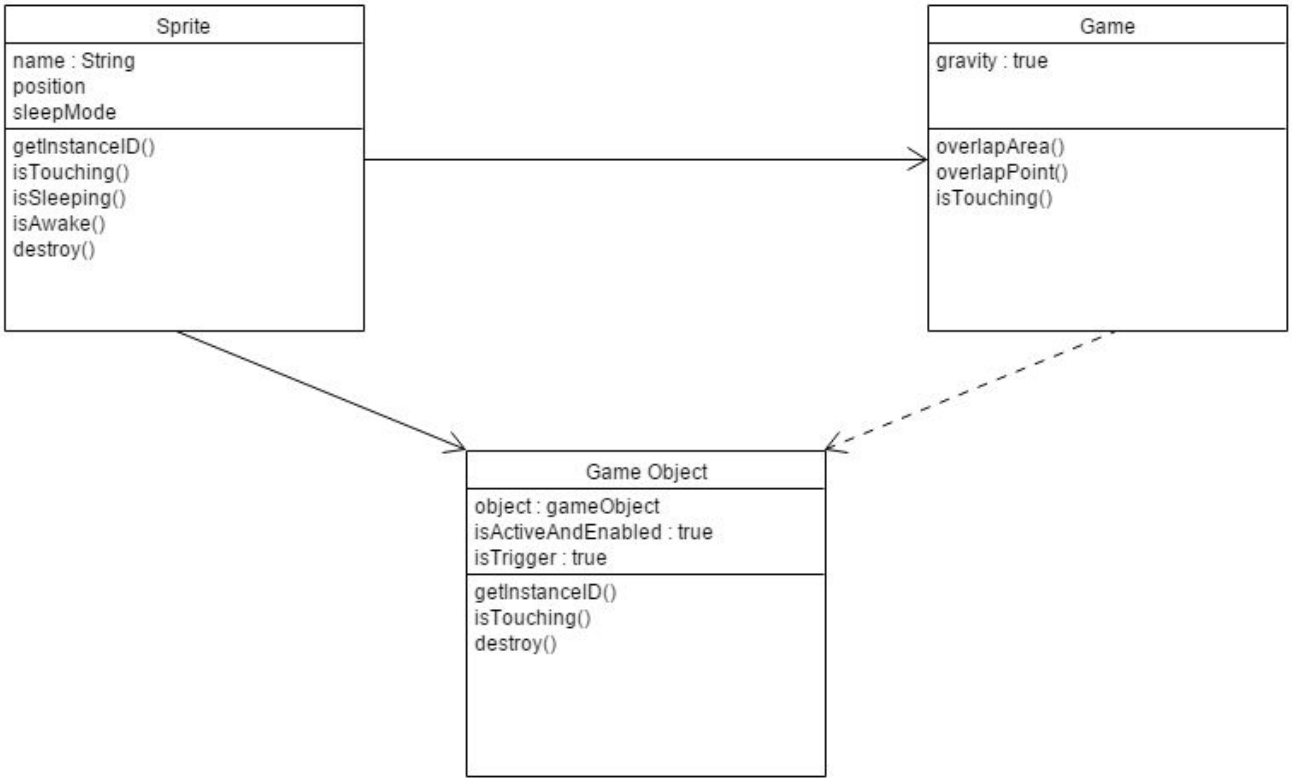


'All Alone?' will make use of the Unity 5 engine, which looks after a lot in the build of the game at the end. The game will be built as a standalone platform - which can be compatible with Mac, Linux and Windows - however we will be focusing on the Windows platform. This will allow us to generate a ".exe" file, with a "Data" folder for all the resources.

We will also be implementing classes from the Unity platform through C#, as detailed in the diagram above. SpriteRenderer and Animation will be used for the visuals, both for the player and any movements made by them, while Collider2D will handle interactions with objects. Rigidbody2D looks after player movements and gravity, simplifying the process for the sprite mapping, while interacting with Collider2D as an attached object.

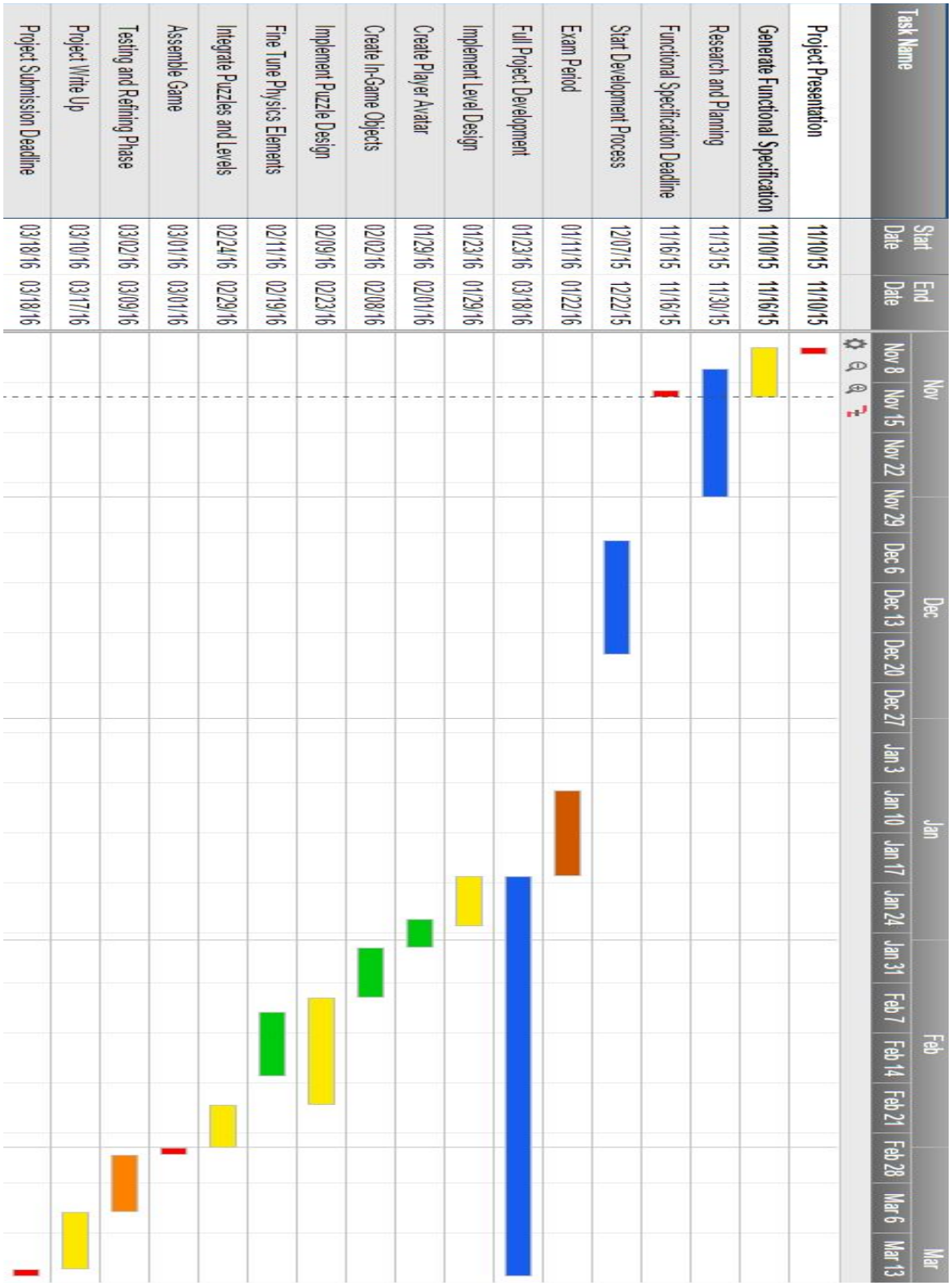
With regards to the objects in the game, Physics2D interacts with Collider2D for the objects the player sprite is interacting with. AreaEffector2D also works with colliders, which we would be using for environmental variables as part of the puzzle aspects in the game.

5. High-Level Design



6. Preliminary Schedule

GANTT Chart



7. Appendix

References

a. Unity Scripting API	http://docs.unity3d.com/ScriptReference/
b. Operating System Market Share	http://www.statista.com/statistics/218089/global-market-share-of-windows-7/
c. C# Programming Guide	https://msdn.microsoft.com/en-gb/library/67ef8sbd.aspx
d. Steam Greenlight	https://steamcommunity.com/greenlight/
e. Kickstarter	https://www.kickstarter.com/
f. Unity Game Engine	https://unity3d.com/
g. Unity Forums	http://forum.unity3d.com/
h. Stack Overflow - Unity	http://stackoverflow.com/questions/tagged/unity
i. Stack Overflow - C#	http://stackoverflow.com/questions/tagged/c%23