****

****

**Assignment Brief – BTEC**

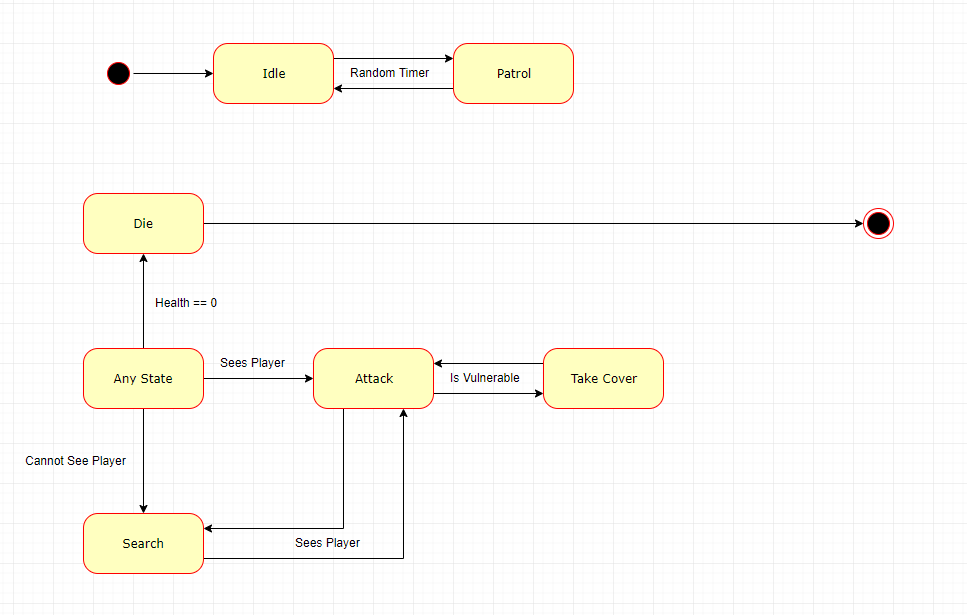
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Programme** | | Level 3 Extended Diploma in Creative Media Production (Games Development) | | | |
| **Unit number(s) and title covered** | | Unit 71: Object-Oriented Design for Computer Games | | | |
| **Assignment number & title** | | Assignment Three: Designing a Game Element | | | |
| **Student name** | | *Lewis Hawkins* | | | |
| **Assessor** | | James Shaun | **Internal Verifier** | *David Matravers* | |
| **Date issued** | | *28/03/2019* | **Submission deadline** | *03/05/2019 at* ***4.30pm*** | |
| **Assessment Criteria** | **To achieve the criteria, the evidence must show that the student is able to:** | | | | **Assessor confirm met** |
| P3 | Review object-oriented modelling with some appropriate use of subject terminology | | | |  |
| P4 | Apply object-oriented modelling techniques to design a game element with some assistance | | | |  |
| M3 | Explain object-oriented modelling with reference to detailed illustrative examples and with generally correct use of subject terminology | | | |  |
| M4 | Apply object-oriented modelling techniques to design a game element to a good technical standard with only occasional assistance. | | | |  |
| D3 | Critically evaluate object- oriented modelling with supporting arguments and elucidated examples, consistently using subject terminology correctly | | | |  |
| D4 | Apply object-oriented modelling techniques to design a game element to a technical quality that reflects near-professional standards, working independently to professional expectations | | | |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Assessor feedback - 1st submission** | | | | | | | | | |
| *Task No* | *Targeted Criteria* | *Met* | *Comment* | | | | | | |
| 1 | P3 |  |  | | | | | | |
| 2 | P4 |  |
| 1 | M3 |  |
| 2 | M4 |  |
| 1 | D3 |  |
| 2 | D4 |  |
| **Did the learner meet the original deadline or agreed extension?** | | | | Yes ☐ No ☐ | | | | | |
| **Assessor signature** | | |  | | | | **Date** | |  |
| **Resubmission authorised?** | | | | Yes ☐ No ☐ | | | | | |
| **New agreed deadline date for submission** *\* must be within 10 days of receiving original assignment back* | | | |  | | | | | |
| **Lead Internal Verifier signature** | | |  | | | **Date** | |  | |
| **Assessor feedback - Resubmission** | | | | | | | | | |
|  | | | | | | | | | |
| **Assessor signature** (resubmission only) | | |  | | **Date** | | | |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Scenario** | | | | |
| After working several months at the internship you gained with Ubiloft, you have progressed well in the developer house and have impressed the lead designer at the studio. She is considering giving you a permanent role, but has so far only seen your technical skills in written form, and needs to be able to see your design skills in action, using industry standard design modelling techniques.  You must demonstrate to her that you are a creative designer, who can efficiently and correctly model a game element of your own design. You must use UML techniques, so that if it is chosen, it can be easily understood by the programming teams so they can implement your designs in possible games such as Observe\_Puppies, and TytanRise. | | | | |
| **Tasks and criteria covered** | | | | |
| **Task 1**  For this task you must develop an overview of UML concepts, relationships, symbols and their use in the design and development of existing products. You should **develop your own UML diagrams (class, sequence and state) of an existing product**, and critically assess how UML can be used as a tool for development of interactive software (games). You must include an explanation of the different relations, models and symbols as well as features that these diagrams contain, such as properties and behaviours.  Your Overview must include:   * **Unified modelling language concepts:** structure (attribute, class); behaviour (activity, event, method); * **Relationships** (aggregation, annotation, composition, depends, generalisation/inheritance) * **Unified modelling language symbols and notation**: classes; relationships; constraints; generalisation * **Unified modelling language diagram types:** structure (class, object); behaviour (statechart); interaction (collaboration, interaction, timing) * **Modelling:** object model (class diagrams); dynamic model (sequence diagrams, statecharts)   [Task Covers P3, M3, D3] | | | | |
| **Evidence you must produce for this task** | | | | |
| Report  UML Diagrams | | | | |
| **Tasks and criteria covered** | | | | |
| **Task 2**  You must now further develop your evidence of UML design but generating a design for a complex game element, using dynamic and static models. You must develop these models with the goal of communicating how your game elements will work for your own original product.  Presentation, annotation and clarity is key for this task, and how these diagrams look will inform the grade to a significant extend, so consider professional standards.  In terms of Game elements, this could include a character class diagram and all associated class relationsships, and that same character used in a dynamic state diagram for a battle event, for example. This may be extended to include behavioural state diagrams.  You should consider the below elements in bold and associated supporting points, as these will assist in the consideration of the development of these game element designs. You must include supporting documentation that comprehenaisvely communicates the goal of the elements and how it achieves the original purpose.  **Game element:** *object, eg sprite, character, vehicle, weapon, rooms, walls, scenery*  **Object attributes:** *properties (colour, visibility, transparency, size, speed, movement); behaviours (mouse*  *events, collision events, keyboard events)*  **Diagram structural relationships:** *objects; instances; inheritance; communication; messaging*  **Diagram event progress**: *objects; properties; events; behaviours; variables; messages; decisions; loops*  **Game element specification:** *documentation, eg unified modelling language (UML), dynamic model,*  *diagrams (structural relationships, event progress)*  [Task Covers P4, M4, D4] | | | | |
| **Evidence you must produce for this task** | | | | |
| UML DIAGRAMS  Supporting Images | | | | |
| **Sources of information** | | | | |
| **Sources of information**  **Textbooks**  Baylis P, Freedman A, Procter N et al – BTEC Level 3 National Creative Media Production, Student Book  (Pearson, 2010) ISBN 978-1846906725  Baylis P, Freedman A, Procter N et al – BTEC Level 3 National Creative Media Production, Teaching Resource  Pack (Pearson, 2010) ISBN 978-1846907371  Gold J— Object-Oriented Game Development (Addison-Wesley, 2004) ISBN 978-0321176608  Makar J – Macromedia Flash MX Game Design Demystified (Macromedia, 2002) ISBN 978-0735713987  Miles R and Hamilton K – Learning UML 2.0 (O’Reilly Media Inc, 2006) ISBN 978-0596009823  Overmars M – ‘Learning Object-Oriented Design by Creating Games’ in Potentials  (the journal of the Institute of Electrical and Electronic Engineers), December 2004-January 2005, Volume 23,  Issue 5, pages 11-13 (available from www.cs.uu.nl/research/techreps/repo/CS-2004/2004-057.pdf)  Rollings A and Morris D – Game Architecture and Design: NRG Programming (New Riders, 2003)  ISBN 978- 0735713634  Swamy N and Swamy N – Basic Game Design and Creation for Fun and Learning (Charles River Media, 2006)  ISBN 978-1584504467  **Websites**  www.cs.uu.nl/research/techreps/repo/CS-2004/2004-057.pdf – article on learning object-oriented design by  creating games, by M Overmars, author of Game Maker software (available from www.yoyogames.com/  make)  www.developer.com/design/ – software development resources and articles  www.devmaster.net/articles/oo-game-design/ – game development encyclopaedia  www.gamasutra.com – respected website for all things game development, sister publication to the respected  print magazine Game Developer; excellent game developer resources  www.macromedia.com/devnet/mx/director/articles/oop\_dir\_flash.html – article on designing and  implementing objects  www.tdan.com/special003.htm – special feature on event progress diagrams | | | | |
| **Student checklist** | | | | **Complete?** |
| Proofread work | | | |  |
| Reference / Bibliography (if applicable) | | | |  |
| All pages attached and numbered – including introduction/conclusion/front sheet | | | |  |
| **Authenticity of Evidence Student declaration** | | | | |
| I certify that the evidence submitted for this assignment is my own.  I have clearly referenced any sources used in the work.  I understand that false declaration of authenticity (i.e. plagiarised work) is a form of academic misconduct and the relevant College procedures will be instigated if I am found to be in contravention of these. | | | | |
| **Student signature** |  | **Date of submission** | 03/05/2019 | |
| **Re-authentication of Evidence Student declaration (for resubmission only)** | | | | |
| **Student signature** |  | **Date of resubmission** |  | |

NB. Students – the assignment starts on the first page **after** these front sheets, i.e. Page 1.

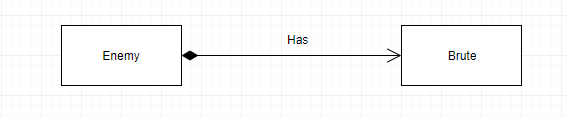
* For your convenience, page numbers have been inserted into the footer. **Please keep them**.
* You may choose to add a contents table (ToC) in this section.
* Please **do** **not use text boxes** for the main body of your written answers.
* Please make sure that images/screenshots are correctly formatted, laid out and labelled. A table of Figures (ToF) may also be added if you wish.
* Make sure you use Page (or Section) Breaks whenever a new page is required. (Rather than adding large numbers of Return/Paragraph characters.) Ensure that new Section breaks continue with correct orientation and correct page numbers.
* Ensure that you have referenced your work throughout, using references in text and that you also have a reference list and full bibliography at the end of the work according to the current **Harvard Referencing** conventions. **Failure to do so will make your work more difficult to authenticate.**

**Task One:**

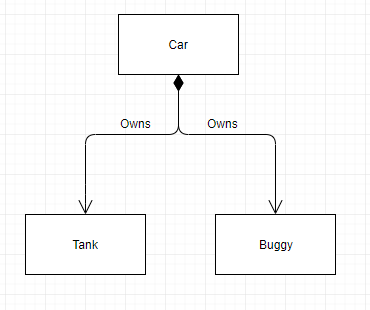
**Relationships**

**Aggregation and Composition** are forms of association which is when two or more classes are linked through a relationship. These relationships do not include parent child (inheritance), but instead it’s when one class “has” or “owns” another.

Aggregation is when a class “has” another which is displayed in the example below.



In the example the Enemy class has the Brute class. Below is an example of Composition.

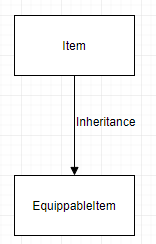


Composition is when a class owns another. Classes are dependent on their owner and will be deleted upon their owner’s deletion.

**Annotations** are used to explain what’s happening in a UML diagram. They can be used to explain relationships, requirements and requirements. If used effectively they can make a diagram easy to read.

**Dependency** is when a class depends on another. A class will become dependent if it’s using another to function. This can happen if an object is instantiated by a class and uses it for information.

**Inheritance:** Inheritance is when a class uses another as a foundation. The parent class stores all relevant information and the subclass takes from that, while implementing its own.



In this example the item class (parent) passes the data to the EquippableItem class which uses it. The data inherited could be name, description, and rarity, since all variations of item will use that. The EquippableItem class will create its own properties and methods such as AttackValue, and Equip();

**Symbols** are used to show the relationships between classes. Using different types eradicates the need for extra annotations, which cleans up the diagram and makes it easier to read.

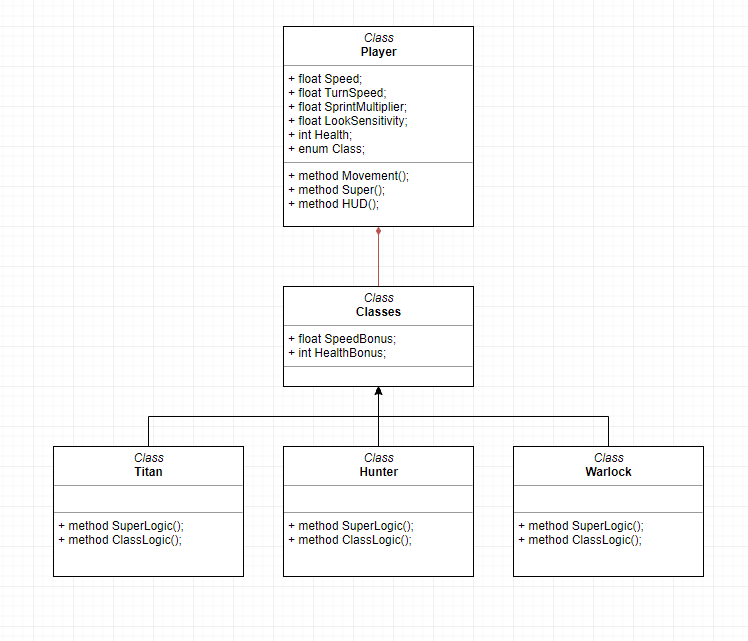
**UML Diagrams**

**Class Diagrams:** A class diagram is used to display how classes interact with each other through displaying messaging and relationships.

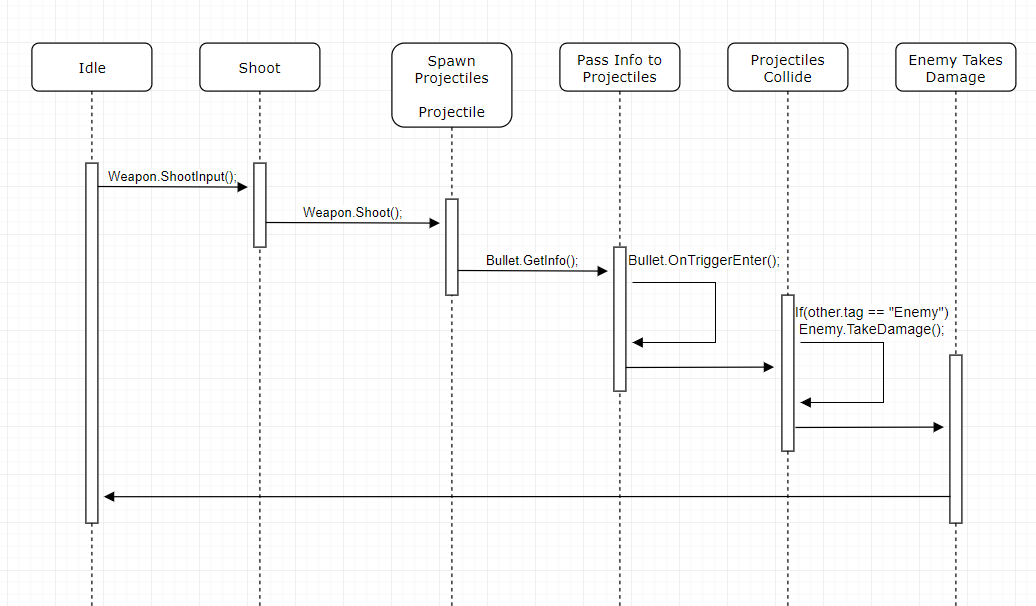
In the diagram below, I used destiny as an example. The player class is interacting with the Classes class, which has 3 subclasses (Titan, Hunter, Warlock). The Classes class has all the properties the subclasses need and passes them down the its children through inheritance. The subclasses then use the properties with their functions to perform the desired way.

Red arrows display Messaging.

Black arrows display Inheritance.

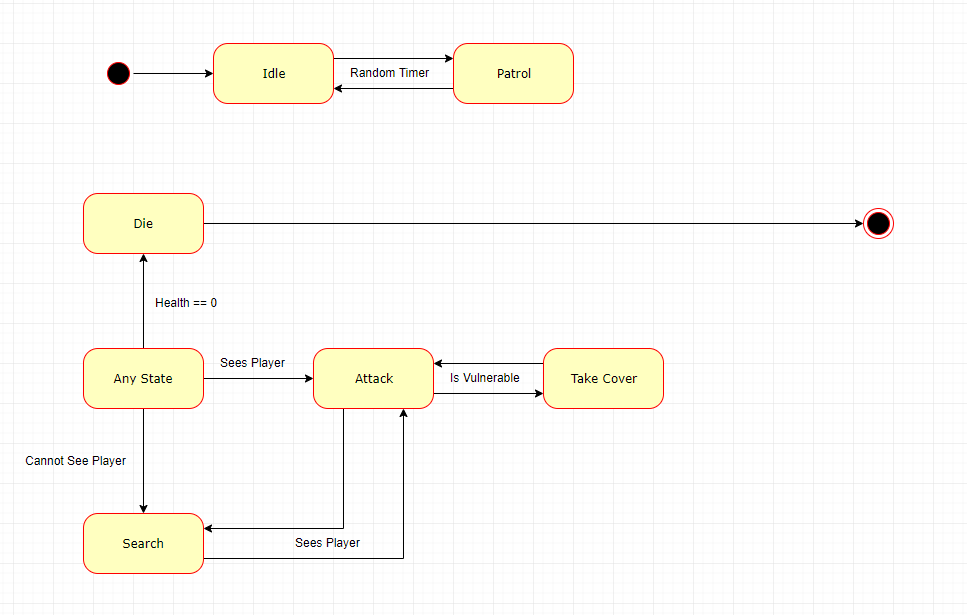


**Sequence Diagrams:** A Sequence Diagram is used to display a scenario and how it plays out. They can be used in games to visualise how menu screens and scripted events.

The diagram below represents a Fusion Rifle in Destiny. The weapon is in idle until ShootInput() is called. Shoot is then called, and the projectiles are instantiated. Once the projectile collides the bullet script checks if it’s an enemy and if so the bullet accesses the enemy script and calls the void TakeDamage().

**State Diagrams:** A State Diagram shows all possible states a game object can be in. It displays them with information on how transitions take place.

The diagram below shows how enemies in destiny react to different situations. It has an any state mode which is used for states that don’t depend on the previous state.



End

Start

**Task Two:**

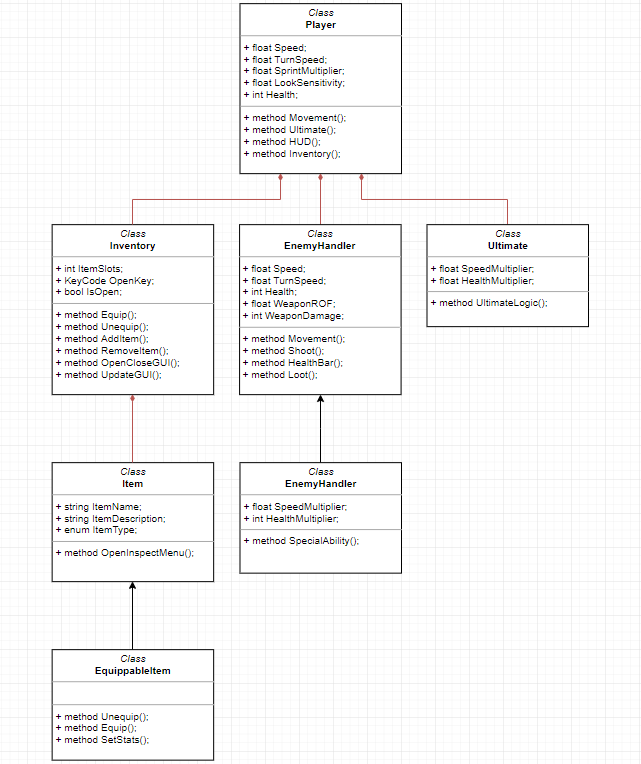
For task two I have decided to use Status Online. This is a game I have developed in which I will use UML diagrams to display some of the core components.

The game is a looter shooter in which the player progresses through missions to gain new weapons and gear. The main character has access to a powerful exo-suit with enhanced movement and strength.

**Class Diagram:**

This diagram displays how the player interacts with the inventory and the enemies. The Ultimate class holds all the code for the Ultimate ability which sends messages to the player.

The EnemyHandler class holds all the basic code for the enemy AI, such as speed, weapon stats, and movement functions. The Brute is a subclass of EnemyHandler and uses it as a foundation. It then adds its own elements to make itself unique.



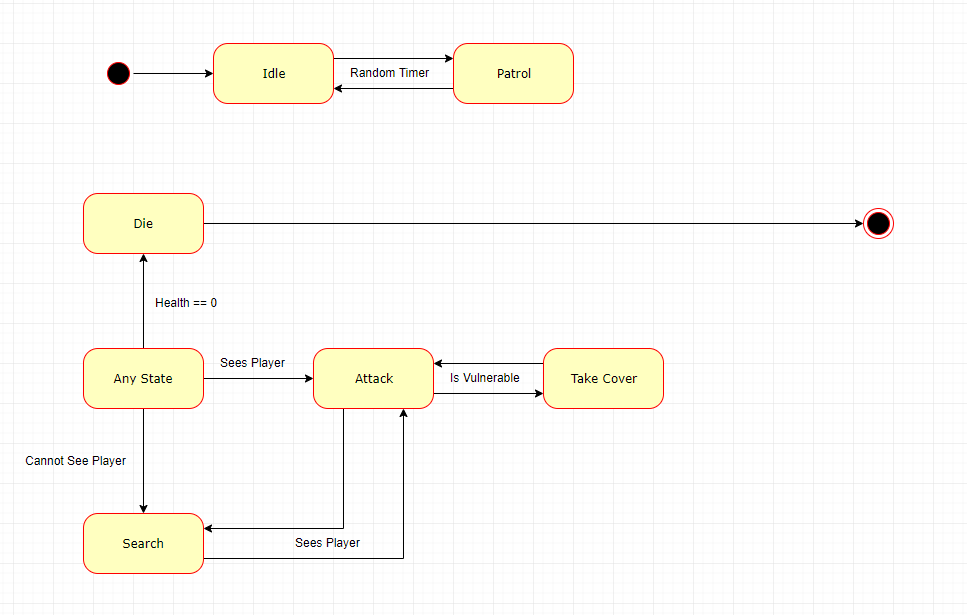
Inheritance

Inheritance

Brute

**State Diagram:**

For the state diagram I used my enemy AI in Status Online. The enemy AI start in a loop of patrol and idle, but quickly switch to attack or search at the first sign of the player through “Any State”. Once the enemy AI transitions to either search or Attack it’s in the loop until it dies or despawns. If the enemy is too vulnerable at its current position, then it seeks cover. Attack is the central state and is used when the enemy is ready to shoot at the player.



**Sequence Diagram:**

This sequence diagram displays how missions are started and replayed in Status Online. The player enters the game through the “Game Start Button” on the main menu. They then start the mission through the mission menu. Once the mission is completed the player can start the next one.

