CGP600 Advanced Games Programming  
AE1 – WBS

Table 1: WBS Dictionary, detailing elements of the WBS Diagram.

|  |  |  |  |
| --- | --- | --- | --- |
| **Level** | **WBS Code** | **WBS Label** | **Definition** |
| 1 | 1. | CGP600 3D Game | Work to Implement the 3D game for the Advanced Games Development assignment. |
| 2 | 1.1 | Inception | Work to begin the project and define the game. |
| 3 | 1.1.1. | Version Control | Set up a central repository for all documentation. |
| 3 | 1.1.2. | Coordinate | Set up a Trello board to help coordinate the development team. |
| 3 | 1.1.3. | Team Info | Store contact details for team members. |
| 3 | 1.1.4 | Brainstorm | Discuss and choose a game idea that fulfils the assignment criteria. |
| 3 | 1.1.5. | Concept Sketches | Produce some sketches that illustrate how the game plays. |
| 3 | 1.1.6. | Concept Map | Make a concept map for the game. |
| 3 | 1.1.7. | Game Definition | Produce a game definition document, including scope and feasibility. |
| 2 | 1.2. | Planning | Work to plan the development of the game. |
| 3 | 1.2.1. | WBS | Start a Work Breakdown Structure diagram/table/dictionary. |
| 3 | 1.2.2. | SDM | Choose and justify a Software Development Methodology. |
| 3 | 1.2.3. | Requirement Definition | Determine game requirements and produce a report detailing functional and non-functional requirements of the game, including levels of importance for individual features. |
| 2 | 1.3. | Analysis | Work to refine the structure of the game ready for the design phase. |
| 3 | 1.3.1. | Domain model | Produce a domain model diagram, showing hypothetical classes in the game and how they interact. |
| 3 | 1.3.2. | Use-cases | Produce use-case diagrams and corresponding use-case tables detailing how the user interacts with the game. |
| 3 | 1.3.3. | Robustness | Produce a robustness diagram that encapsulates interaction between the user and the game, as detailed by use-cases. |
| 3 | 1.3.4. | Sequence | Produce a sequence diagram that shows step-by-step game execution. |
| 3 | 1.3.5. | Classes | Produce a class diagram that details classes and their relationships, as identified from domain, use-case, and robustness diagrams. |
| 3 | 1.3.6. | Decomposition | Produce a decomposition diagram that details game processes. |
| 2 | 1.4. | Design | Work to determine what will go into the code prior to the implementation phase. |
| 3 | 1.4.1. | Framework | Windows, DirectX, and game start up and shutdown, and main loop. |
| 3 | 1.4.2. | Input Manager | Central input management. Start with basic Windows input events, explore DirectInput options. |
| 3 | 1.4.3. | Resource Manager | Central resource management (memory & assets). |
| 3 | 1.4.4. | Scene Manager | Central FSM for transition between game scenes. |
| 3 | 1.4.5. | Splash Scene | Front-end splash scene. |
| 3 | 1.4.6. | Front-end Scene | Front-end main menu and sub menu scene. |
| 3 | 1.4.7. | Game Scene | Core game logic. |
| 3 | 1.4.8. | Menus | Reusable menu system, for all front-end and in-game menus. |
| 3 | 1.4.9. | Game Objects | Abstract base class for all objects. |
| 3 | 1.4.10. | Hover-tank | Extends game object class for the Hover-tank vehicle. |
| 3 | 1.4.11. | Player Tank Control | Extends the Hover-tank class with a user control method. |
| 3 | 1.4.12. | AI Tank Control | Extends the Hover-tank class with an AI control method. |
| 3 | 1.4.13. | Static Obstacles | Solid game objects within the environment that do **not** move when Hover-tanks collide with them. |
| 3 | 1.4.14. | Movable Obstacles | Solid game objects within the environment that **do** move when Hover-tanks will collide with then. |
| 3 | 1.4.15. | Collectables | Non-solid game objects within the environment that de-spawn only when user Hover-tanks collide with then. |
| 3 | 1.4.16. | HUD | Screen overlay that shows collected/remaining collectables in the environment. |
| 3 |  |  |  |
|  |  |  |  |
| 2 | 1.5 | Implement | Work to write the actual code. |
| 2 | 1.6 | Test | Work to test the game. |
|  |  |  |  |