**Game Specification Form Student ID: 000646118 / sklv77 Level 4**

|  |  |
| --- | --- |
| **Marking Criteria** | **Describe how your game matches the criteria** |
| **Game design (10%)** | |
| Game Goals: | Find all 10 treasures. |
| Game Type: | First person, treasure seeking, survival/hiding from spiders game. |
| **Core development (30%)** | |
| Game scene (visual representation [2D, 2.5D or 3D], internal data structure): | 3D terrain (Unity). |
| Game flow / game progression (e.g., navigation, screen scrolling, levels): | Large map, stays the same for all levels, treasure also remain in the same places. Each level gets progressively more difficult: the player sprint speed decreases, the distance from the player that the spiders can start running increases, the walk speed of the spiders increases and the number of spiders generated per unit time increases. |
| Game interaction (e.g., action detection and response generation): | Walking into treasure picks it up.  Pink capsules are power ups and give the player the ability to boost, boosting teleports 100m in the direction that it is looking.  Spiders are the enemy in the game, touching them will cause the player to lose a life (the player initially has 10 lives). Spiders will only see the player within a 50m radius, if they can see the player they will walk towards them (spiders have the NavMeshAgent component). If the spider is closer to the player it will run, and then attack when very close. Spiders can be clicked on when they are close to stun them.  I have made a limited-duration power-up, hiding, which causes the spiders not to see the player, the spiders will walk around randomly until the 10 second hiding time is up; then if they can see the player they will chase again. If they cannot see the player after 10 seconds they will check every 5 seconds if they can see the player, otherwise they walk around randomly until the player is too far away (they are destroyed). |
| Game object (e.g., use of sprite, 3D objects, animation, multimedia): | I have used a free spider prefab and activated some of the animations depending on the distance to the player (walking when it can see the player, running when its closer, attacking when it’s very close and dying when it hits the player). A scary theme song also plays when many spiders are in pursuit of the player (and the hiding time has run out). |
| **Game mechanics (30%)** | |
| Game rules / logics: | Avoid spiders, getting hit by one will make the player lose a life, the player has 10 lives for each level. The player dies if in water that’s too deep. Use boosts to escape spiders/get up to places and importantly to find treasure. |
| Game challenges: | Find all 10 treasures to complete the level. (If you want to know where all the treasures are, go to the prefabs folder and drag into the scene and you will be able to see where I have placed them all). Avoid spiders and the river to survive. The player can press ‘h’ to hide, which means the spiders can’t see you for 10 seconds, so get to somewhere they can’t see you fast! If there are boost power ups it is to signify that the player is near a treasure where a boost is required (to jump up onto a building). To use the boost, point the camera so the player is looking above the place you want to jump, then press ‘b’. One place you have to use this is to get on top of the cathedral (there is a door on the top that goes to the crypt). |
| **Good use of game engine (15%)** | |
| Choice (pyGame, Unity): | Unity. |
| User input (keyboard, mouse, joystick): | Keyboard and mouse. (Keys ‘w’, ‘a’, ’s’, ‘d’ to move, ‘space bar’ to jump, ‘shift’ to sprint, ‘h’ to hide, ‘b’ to boost). (move mouse to look around, click to stun spiders.) |
| Game object interaction (e.g., event triggering, collision detection): | Collision detection with trees/terrain/buildings/spiders.  Event triggering:  OnControllerColliderHit: hitting spiders, picking up power ups.  RayCast: for boosts.  OnMouseDown: clicking to stun spiders.  OnTriggerEnter: when the player walks into the door on the top of the cathedral, they are teleported to the crypt. |
| Incorporate multimedia content: | If there are many spiders in pursuit of the player, and the player has pressed ‘h’, when the hiding timer runs out scary music plays. |
| Other features used (e.g., asset, incorporation of external libraries): | Standard assets, terrain tools, cathedral, castle, door, crypt, old house, bridge, water assets, spider, pillars, coins. Walls/lamps/floor assets which I have made buildings from for the science site. (See README file for complete list.) |
| **Demonstrate creativity (15%)** | |
| Game economy (e.g., support to game type, game feedback, game difficulty): | If the player is avoiding many spiders over a long period of time, the game will generate more spiders and increase their speed. If the player is losing lives fast, less spiders will be spawned. If a player hasn’t lost a life for a long time, the rate will increase, if a player is losing lives fast the rate will decrease. |
| Advanced Interaction (e.g., game physics, object tracking, steering behaviour): | Clicking on the spiders stuns them. The door on top of the cathedral teleports the player to the crypt. Boosts teleport the player approximately 100m in front of the player. I have placed a NavMeshAgent on each spider to track the player, colliders to register if the player hits objects/spiders. Spiders are spawned by: adding the players position to the direction the player is multiplied by 50, then I randomly select a vector in the sphere of radius 50, add them together, then find the closest point to it on the NavMesh to spawn a spider. This gives the effect of spiders spawning in front of the player, which works well if the player is running away. |
| **Game optimisation and configurability (50%) [For Level 4 Students Only]** | |
| Include optimisation to enhance game performance (e.g., game related functions, game scene and objects, interaction, rendering, media content): | To reduce computation, I have structured the scripts so there is only the GameManager script and the SpiderMovement Scripts using the update function. On top of this I only call OnGUI from one script at a time (I do this by separating the different GUI elements into different files and enabling/disabling them with the GameManager script). I have used event listeners where possible for example: OnMouseDown in the spider movement script, this avoids having an if statement in an update function.  To avoid ever using FindGameObject or using any function for searching for GameObjects since these operations are very slow I have made public GameObject variables (cached) which I assign before running the game. In the “Start” function of some scripts I have also cached components (where I have to get a component of a GameObject), since this is a fairly slow function.  I have set all stationary GameObjects to static and baked the scene so that the only lighting that needs to be computed while the game is running is for that of moving objects.  Since many spiders could spawn and the player could run away and they would still use up compute time, but be nowhere to be seen. To avoid this I have used a list that keeps track of them all in GameManager, and relatively irregularly (every 5 seconds using Invoke recusively, to avoid doing this in the update function) I destroy the GameObject for spiders that are far away from the player. I also use a coroutine to re-enable the NavMeshAgent on spiders after they have been stunned.  Where possible, for efficiency, I have used “Convert.ToInt32(\*some boolean expression\*)” to avoid using an if statement.  Where possible I have used box or sphere colliders, since they are less computationally intensive.  All of the textures in the project have MipMaps enabled. I have also enabled GPU instancing on the materials.  I have changed the audio used for the FirstPersonCharacted footsteps/jump etc to ADPCM audio compression for higher compression. For my SpiderAttackTheme mp3 I have set it to ‘Compressed in Memory’ to reduce memory. |
| Make the game flexible to support making changes (e.g., game scene and objects, game flow / progression): | All my scripts are placed into a scripts folder in the Assets folder, and then organized into further folders for convenience. I have included a prefabs folder for GameObjects that will be instantiated by script, so others can be easily added. The GameObjects in the scene are organized into empty GameObjects so it doesn’t appear cluttered.  I have made a levels menu UI Panel GameObject and a script, “LevelsMenu” which levels can very easily be added too. It is also very easy to extend the map by adding more terrains. The difficulty can be easily tweaked, or the function that adapts the difficulty to the skill of the player (AdaptDifficulty in GameManager).  An exampled of an addition I would have made if I had more time is: on higher levels, I would’ve added a flock of boids that fly around the map, and if they can see you, they attack/spawn more spiders (as I think this would make the game more fun). Additions like this would be easy to put into the game, as the each boid could have its own script for its behaviour (defining simple rules based on what is in a small radius of it for what movement to do), and only small changes to the GameManager script would need to be made (how many to add, and on which levels etc). Another example of a simple idea that could be added to support making changes to do with progression, is having a snowy level where if you stay in snowy parts of the map for too long (use my recursive invoking start timer function I have used in the hiding and notHiding scripts) it would take away a life. Another good way to extend the game would be to have total coins earned on the menu, and allow the player to purchase new abilities eg. more boosts/longer hiding (would be displayed on a new menu panel). |