|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| VERSIÓN | IMPLEMENTACIÓN | RESTRICCIONES | EVIDENCIA | AVANCES |
| 1 | OpenGL: Primer acercamiento a movimiento aleatorio de partículas. | Interfaz gráfica limitada, movimiento sesgado a un aleatorio parcializado. |  | **¿La partícula se moverá con ‘libre albedrío’? 20 %**  **¿Cómo implementar los obstáculos? 10%** |
| 2 | JavaScript + paper.js:  Movimiento aleatorio de varias partículas basado en 4 direcciones(cardinales). | El sistema implementado no permite un movimiento fluido y da la sensación de vibración. |  | **¿La partícula se moverá con ‘libre albedrío’? 30%**  **¿Que conoce y que desconoce la partícula? 10%**  **¿Como intercambiarían información entre partículas? 10%** |
| 3 | JavaScript + paper.js:  Movimiento aleatorio de varias partículas basado en vector direccional.  Primera implementación de obstáculos. | Fallos en la acción después del contacto con el obstáculo.  Fluidez en el movimiento no es completa. |  | **¿La partícula se moverá con ‘libre albedrío’? 40%**  **¿Cómo implementar los obstáculos? 20%** |
| 4 | JavaScript + paper.js:  Implementación de cronómetro y ciclos de actividad. Exploración durante 8 segundos e inactividad durante 8 más. | Las partículas terminan la exploración y reaparecen en el punto de origen, falta implementar el regreso. |  | **¿La partícula se moverá con ‘libre albedrío’? 40%**  **¿Cómo implementar los obstáculos? 30%**  **¿Cuál es el ciclo de actividad de las partículas? 20%** |
| 5 | JavaScript + paper.js:  Primera conexión de la base de datos manejando la información del mundo.  Actualización de datos en tiempo real. | El manejo de datos no es masivo, se deben realizar pruebas con más volúmenes de datos. |  | **¿Que conoce y que desconoce la partícula? 20%**  **¿Como intercambiarían información entre partículas? 20%**  **¿Cuál es el ciclo de actividad de las partículas? 20%** |
| 6 | Unity 3D:  Segunda implementación de obstáculos, obstáculo que se debe rodear.  Prueba con varios orígenes. | La implementación de la colisión no es perfecta. |  | **¿Cómo implementar los obstáculos? 40%** |
| 7 | Unity 3D:  Prueba con varios orígenes y obstáculo de rodear. | No hay |  |  |
| 8 | Unity 3D:  Implementación de obstáculo tipo 2, disminuir la velocidad de la partícula cuando lo atraviese. | No hay |  | **¿Cómo implementar los obstáculos? 45%**  **¿Cuál es el ciclo de actividad de las partículas? 40%** |
| 9 | Unity 3D:  Reconocimiento del camino y distancia recorrida por cada partícula. Visualización gráfica. | No hay |  | **¿Que conoce y que desconoce la partícula? 30%**  **¿Cuál es el ciclo de actividad de las partículas? 40%** |
| 10 | Unity 3D:  Dejar señales de encuentro con obstáculo | Las señales representantes del encuentro se deben conectar a la base de datos. |  | **¿Que conoce y que desconoce la partícula? 40%** |
| 11 | Unity 3D:  Movimiento aleatorio de una partícula por el mundo. | No hay |  | **¿La partícula se moverá con ‘libre albedrío’? 30 %** |
| 12 | Unity 3D:  Movimiento aleatorio de varias partículas por el mundo.  Reconocimiento de partículas alrededor. | No hay |  | **¿La partícula se moverá con ‘libre albedrío’? 40 %** |
| 13 | Unity 3D:  Intercambio de datos Unity-Firebase con base de datos online en tiempo real: implementación de datos relacionados a:   * Colisiones entre partículas. * Posición de la colisión. * Ciclo activo durante la colisión. | Paso a seguir:  Agregar progresivamente todas las interacciones entre las partículas y analizar los datos para modificar el comportamiento de estas.  Indicios de emergencia. |  | **¿Que conoce y que desconoce la partícula? 50%**  **¿Como intercambiarían información entre partículas? 40%** |