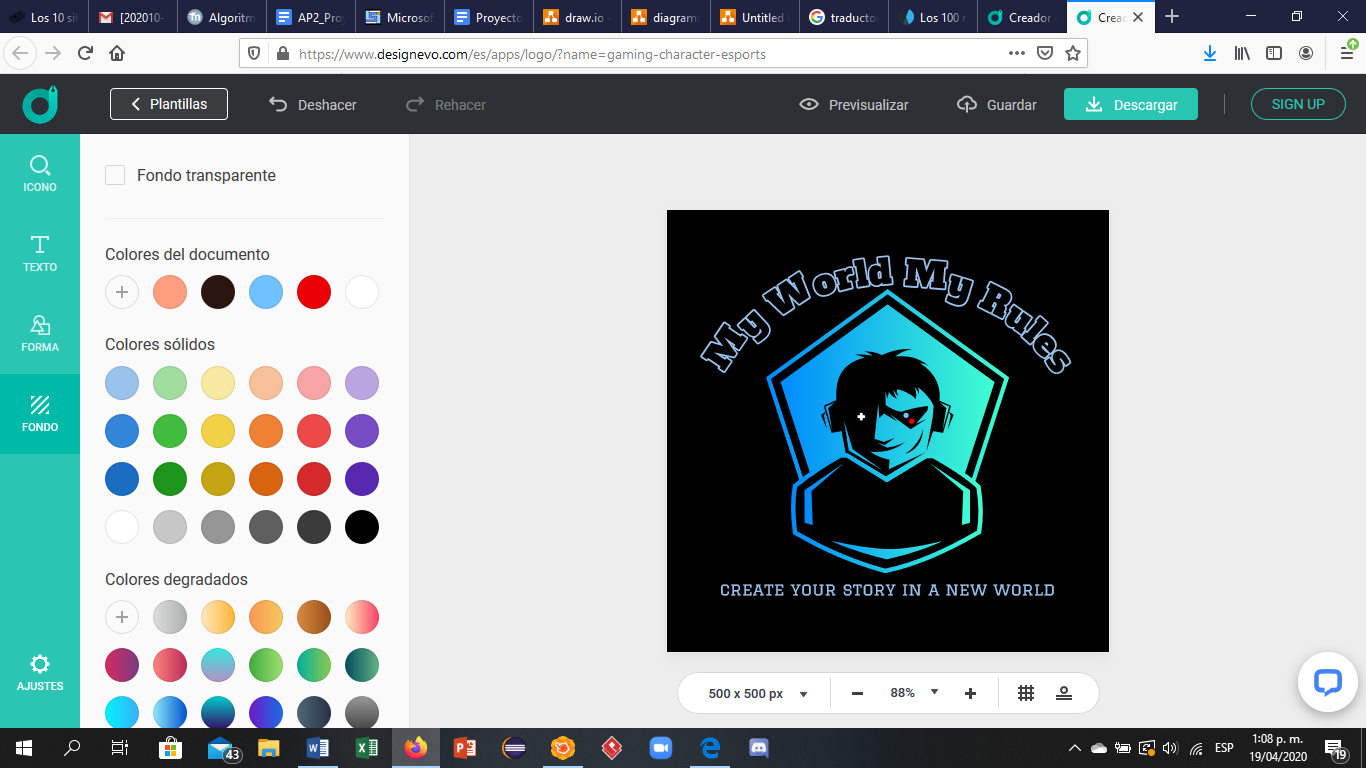
**Algoritmos y Programación II**

**2020-1 Segunda Entrega Proyecto Final de curso**

|  |  |
| --- | --- |
| **Nombre:**  **Alexander Samaca Burbano**  **Jhon Sebastian Ijaji Ortiz**  **Sebastian Villa Avila** | **Codigo:**  A00362108  A00362423  A00361589 |
| **Profesor:**  **Juan Manuel Reyes** | |

1. **Nombre del proyecto:**



1. **Requerimientos Funcionales**:

El sistema estará en capacidad de:

* Guardar partidas en ciertos puntos del nivel (checkpoints) para así poder reanudar la partida incluso luego de cerrar el juego, hay que tener en cuenta que los jugadores podrán borrar sus partidas si así necesitan o requieren.
* Permitir a los personajes moverse en un plano en 2da dimensión, teniendo lo siguientes movimientos: saltar, avance izquierdo, avance derecho, bajar una plataforma y/o agacharse.
* clasificar puntuaciones de acuerdo al usuario, de manera que clasificara los puntajes de mayor a menor creando un ranking para los mejores jugadores
* registrar un nuevo usuario con un Nickname y una foto como avatar de tal manera que se añadirán al juego los nuevos avatares para empezar una aventura por el juego
* Tener un 2do jugador para poder completar todos los niveles, haciendo necesario la cooperación mutua para el avance en los niveles.
* generar un reporte de las ultimas partidas que realizado el jugador, para que este pueda ver la cantidad de partidas y el puntaje obtenido en cada una.

1. **Requerimientos no Funcionales:**

* El sistema será capaz de utilizar persistencia (serializacion) para guardar las partidas jugadas por un equipo y que este pueda retomar su partida en el nivel que había quedado.
* El sistema utilizara la búsqueda binaria para encontrar el nickname ingresado por el usuario y tomar los datos de este para retomar su partida
* El sistema utilizara excepciones para alertar al usuario al momento de realizar una acción invalida o al detectar falta de información en algún campo necesario para el juego
* El sistema guardara la puntuación de los usuarios en archivos de texto usando persistencia
* El sistema utilizara los diferentes métodos de ordenamiento para la tabla de clasificaciones a nivel global y a nivel de usuario
* El sistema estará en capacidad de almacenar los usuarios registrados utilizando listas enlazadas
* El sistema contara con interfaces graficas capaces de interactuar con el usuario para darle una experiencia única
* El sistema usará arboles binarios como método para el desplazamiento entre niveles, dado que en el juego las diferentes acciones de los jugadores conllevaran a diferentes líneas del mapa.

1. **Diseño de pruebas**

**ESCENARIO:**

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Clase** | **Escenario** |
| SetUp1 | User | Seyerman1 1645800  Seyerman2 2000000  Seyerman3 5632100 |

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Clase** | **Escenario** |
| SetUp2 | Registry | Stian, 180000  Gevorah 200000  Alex 1500000  Syerman 1645800  Tjking 120000  Destroyer 621000 |

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Clase** | **Escenario** |
| SetUp3 | User | stian, Imagen |

**CASOS:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** esta prueba se encarga de probar cada método de la clase Resgistry. Es decir, se encarga de probar que se añada el usuario que se ordenen los puntajes y que muestre el historial de jugadas y puntajes del usuario. | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado** |
| Registry | addUser | SetUp3 | Stian, (imagen del avatar) | El registro correcto de un nuevo usuario al juego y la carga de pantalla de niveles para empezar a jugar. |
| Registry | addUser |  | Campos en blanco | Lanza una excepción y un mensaje de alerta que no le permite al usuario avanzar hasta que llene los campos correctamente |
| Registry | addUser | SetUp3 | Stian, (imagen del avatar) | Lanza una excepción y un mensaje de alerta que no le permite al usuario avanzar dado que ya hay otro usuario con el mismo Nickname, por tanto deberá cambiarlo para poder seguir |
| Registry | organiceScore | SetUp2 | Stian, 180000  Gevorah 200000  Alex 1500000  Syerman 1645800  Tjking 120000  Destroyer 621000 | El listado de puntajes ordenados de mayor a menor para mostrarlo en un tabla y mostrar un ganador el cual tendrá el mayor puntaje y la mejor historia en el mundo. |
| Registry | record | SetUp1 | Lista de partidas con puntajes | Mostrarle al usuario sus ultimas partidas y los puntajes obtenidos en esta también se guradara en un archivo de texto de tal forma que esta información pueda ser obtenida después |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** el objetivo de esta prueba es verificar que los usuarios se creen correctamente de tal forma que puedan convertirse en un avatar en el juego | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado** |
| User | User | SetUp1 | Nickname, imagen | Garantiza la creación de un usuario correctamente. Es decir que el usuario cuente con todos sus atributos |
| User | getNickName |  | Ninguno | El nickname del jugador para utilizarlo en la tabla de puntajes y demás funciones del juego |
| User | setImage |  | Nueva imagen para el avatar | Actualiza correctamente la imagen del avatar, dándole al usuario la posibilidad de cambiar su estilo en cualquier parte del juego |

**NOTA:** las clases restantes son encargadas de los acciones y movimientos de los avatares en el juego de tal forma que en este momento de la implementación no se le puede hacer diseño de pruebas ya que está en proceso de desarrolla para completar el juego.