



# Anime Maze

**Nombre: José Rafael Pérez Rivero**

**Grupo: C-112**

Link del proyecto: <https://github.com/joserafael0160/Maze-Runners>

# Ideas sobre el proyecto

- Elección de tecnologías
- Elección de la tema del proyecto, historia y música
- Proceso de creación
- Diseño



# Tecnologías Utilizadas

- **C#:** Lógica del juego y algoritmos.
- **Blazor .Net 8.0:** Framework principal para la aplicación web.
- **Tailwind CSS:** Estilizado de componentes.
- **JavaScript Interop:** Integración con APIs del navegador
- **LibreSprite:** Para el diseño de los personajes, paredes, obstáculos, trampas, la meta, etc



# Proceso de Creación

1. Laberinto definido a mano
2. Jugadores básicos
3. Movimientos
4. Condición de victoria
5. Trampas
6. Página de selección de Personajes
7. Página de selección de número de jugadores
8. Página de selección de tamaño del mapa
9. Página principal
10. Estilos



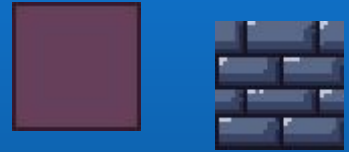
# Diseño

- Estilo
- Pixel art Kawaii

## - Botones para jugar



## - Paredes y camino



## - Meta



## - Obstáculos



## - Trampas



## - Sonido



## - Otros Elementos



## Personajes



# Flujo del proyecto /home

Fondo de  
partículas  
dinámico

Activar  
Música

Inicio  
Rápido

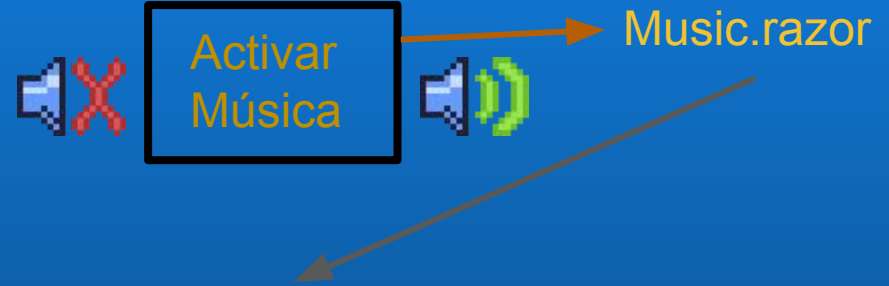
Historia

Tarjeta de  
presentación

Muñeco que se mueve  
junto con el ratón



# Flujo del proyecto



```
<div class=volume-control>  
  <button toggle>
```

```
<audio id="audioPlayer">
```

```
playlist = ruta de canciones
```

```
OnAudioEndeed maneja el evento de fin de  
reproducción para pasar a la siguiente  
canción
```





# Flujo del proyecto

Fondo de partículas  
dinámico

```
<div coleccion de particulas>  
  <div particula>
```

## Generación Aleatoria de Partículas

Se usa un bucle `for` para generar 30 partículas con posiciones iniciales aleatorias dentro del contenedor.

Las posiciones `top` y `left` de cada partícula se determinan usando `Random.Shared.Next(0, 100)` para distribuirlas aleatoriamente en el contenedor.

Muñeco que se mueve  
junto con el ratón

```
<figure>  
  <image>
```

## Actualización de la posición

La posición de la figura se actualiza continuamente alejándose del cursor del mouse mediante un método

`UpdateMousePosition`, que se llama desde JavaScript.



# Flujo del proyecto /History



# Flujo del proyecto

[← Volver](#) Historia

## Header

`<a href="home"> ← Volver`

`<h1>` encabezado (gradiente)

Componente derecho, Parametros:

- Title
- ShowBackButton determina si se muestra o no
- RightComponent un fragmento de renderizado opcional para añadir contenido adicional



Inicio del juego

Link a /MazeSizeSelection



# Flujo del proyecto

## /Models/HeroImage.cs

```
class HeroImage
{
    Down
    Up
    Left
    Right
    HeroImage(string name)
    {
        Down = $"images/Characters/{name}/Down.png"
        ...
    }
}
```

## /Models/Power.cs

```
class Power
{
    Name
    Description
    Cooldown
    CurrentCooldown = 0
    Action<Player, List<Player>> Action
    Power(name, description, cooldown, action)

    void Use(Player player, list<Player> players)
    {
        if (CurrentCooldown == 0)
            Action(player, players)
            CurrentCooldown = Cooldown
    }
    void ReduceCooldown()
    {
        if (CurrentCooldown > 0)
            CurrentCooldown--
    }
}
```



# Flujo del proyecto /Models/TemporaryEffect.cs

## /Models/Hero.cs

```
class Hero
{
    Name
    Description
    Speed { set {entre 0 y 5}}
    Health { set {entre 0 y 100}}
    Attack { set {entre 0 y 40 (En caso de que Name = Saitama
puede tener 100)}}
    HeroImage Image
    Power Power
    Hero(name, description, speed, health, attack, power)
}
```



```
class TemporaryEffect
{
    Name
    Action<Player> ApplyEffect
    Action<Player> RemoveEffect
    TurnsRemaining

    TemporaryEffect(name, applyEffect,
removeEffect, duration)

    void Apply(Player player)
        ApplyEffect(player)

    void Remove(Player player)
        RemoveEffect(player)

    void DecrementTurn()
        if (TurnsRemaining > 0) TurnsRemaining--
}
```

# /Services/TurnManager.cs

TurnManager:

Variables:

- CurrentPlayerIndex = 0
- CurrentPlayer:
  - Si hay jugadores en PlayerData.Players:
    - Retornar el jugador en CurrentPlayerIndex
  - Si no, retornar null
- MovesLeft

Constructor:

- Llamar a ResetTurns()

Función NextTurn():

- Si CurrentPlayer no es null:
  - Decrementar MovesLeft en 1
  - Si MovesLeft  $\leq$  0:
    - Llamar a ApplyEffects() en CurrentPlayer
    - Reducir el cooldown del poder del héroe seleccionado (si lo hay)
      - Incrementar CurrentPlayerIndex (y usar módulo para ciclar entre jugadores)
  - Establecer MovesLeft a la velocidad del CurrentPlayer

Función ResetTurns():

- Establecer CurrentPlayerIndex a 0
- Para cada jugador en PlayerData.Players:
  - Limpiar TemporaryEffects
  - Establecer Position a InitialPosition
  - Establecer Health al valor del héroe seleccionado (o 0 si no hay héroe seleccionado)
    - Establecer Speed al valor del héroe seleccionado (o 1 si no hay héroe seleccionado)
    - Establecer Attack al valor del héroe seleccionado (o 0 si no hay héroe seleccionado)
      - Si CurrentPlayer no es null y tiene un héroe seleccionado:
        - Establecer MovesLeft a la velocidad del héroe seleccionado
      - Si no:
        - Establecer MovesLeft a 0



# /Services/ValidationService.cs

ValidationService:

Variables:

- game: Labyrinth = LabyrinthData.Game

Función IsValidMove(playerRow, playerCol):

- Verificar si playerRow y playerCol están dentro de los límites del laberinto
- Verificar si el tipo de celda es Road, Exit, Obstacle con Health = 0, o Trap
- Retornar true si todas las condiciones son verdaderas, de lo contrario retornar false

Función IsWinningMove(playerRow, playerCol):

- Verificar si el tipo de celda en playerRow y playerCol es Exit
- Retornar true si es Exit, de lo contrario retornar false

Función CanAttack(playerRow, playerCol):

- Obtener la celda en playerRow y playerCol
- Verificar si playerRow y playerCol están dentro de los límites del laberinto
- Verificar si la celda es un Obstacle o si hay un jugador en la posición
- Retornar true si alguna condición es verdadera, de lo contrario retornar false

Función GetCell(playerRow, playerCol):

- Retornar la celda en playerRow y playerCol del laberinto

Función GetPlayerAtPosition(row, col):

- Verificar límites del laberinto
- Si la posición está fuera de los límites, retornar null
- Buscar y retornar el primer jugador en la posición (row, col)
- Si no se encuentra ningún jugador, retornar null

Función ResetValidationGame(newGame):

- Establecer game a newGame

# /Models/Player.cs

```
enum Direction { Up, Down, Left, Right}
```

```
class Player
{
    (RowPosition, ColPosition) Position
    (RowPosition, ColPosition) InitialPosition
    Hero HeroSelected
    bool HasWon, int Health, Speed, Attack
    Direction FacingDirection
    List<TemporaryEffect> TemporaryEffects
    Player (position)

    void InitializeStats() Health = HeroSelected.Health;
    ...
}
```



```
bool MovePlayer(direction)
{
    (newPositionRow, ...Col) = (Position.RowPosition,
    ...Col)
    switch (direction)
    {
        case "up":
            newPositionRow--;
            FacingDirection = Direction.Up
            ...
    }
    if (ValidationService.IsValidMove(newPositionRow,
    ..)
    {
        Position = newPositionRow
        if (ValidationService.IsWinningMove
        (newPositionRow, ...))
        {
            HasWon = true
            return true;
        }
        TurnManager.NextTurn()
    }
    return false
}
```



# Continuación

## /Models/Player.cs

```
void ApplyEffects()
{
    foreach(effect in TemporaryEffects)
        effect.DecrementTurn()
    if (effect.TurnsRemaining <= 0)
        effect.Remove(this)
    TemporaryEffects.Remove(effect)
}

void AddTemporaryEffect(effect)
{
    TemporaryEffects.Add(effect)
    effect.Apply(this)
    if (Health <= 0)
        Position = InitialPosition
        InitializeStats()
}

void ResetHasWon()
{
    HasWon = false
}
```

```
void AttackTarget()
{
    (row, col) attackPosition = Position;
    switch FacingDirection
    {
        case Direction.Up:
            attackPosition.row--;
            ...
    }
    if (ValidationService.CanAttack(attackPosition))
    {
        cell = ValidationService.GetCell(attackPosition)
        player = ValidationService.GetPlayerAtPosition(attackPosition)

        if (es un obstaculo y su vida > 0)
        {
            cell.Obstacle.TakeDamage(this.Attack)
            if (cell.Obstacle.IsBroken())
            {
                cell.Obstacle = null
                cell.Type = Labyrinth.CellType.Road
            }
        }
        else if (es un jugador)
        {
            player.Health -= this.Attack
            if (player.Health <= 0)
            {
                player.Position = player.InitialPosition
                player.InitializeStats()
            }
        }
        TurnManager.NextTurn();
    }
}
```

# Flujo del proyecto

## /Models/Trap.cs

```
class Trap
{
    Name
    Description
    Image
    Action<Player> Activate = (player) =>
    {
        if (player no tiene ese efecto)
            player.AddTemporaryEffect(new TemporaryEffect
                name,
                p => {} Aplicar efecto,
                p => {} Remover efecto,
                0      Duración de turnos del efecto
            ))
    }
}
```

## /Models/Obstacle.cs

```
class Obstacle
{
    Name
    Image
    Health
    MaxHealth
    Obstacle (health, name, image)

    void TakeDamage(damage)
    if (Health > 0)
        Health -= damage;
        if (Health < 0) Health = 0

    bool IsBroken() return Health <= 0

    void Reset() Health = MaxHealth
}
```



# Flujo del proyecto

## /Models/Labyrinth.cs

```
enum CellType {Road, Wall, Trap, Obstacle, Exit}
```

```
class Cell
{
    CellType Type
    Trap Trap
    Obstacle Obstacle
    Cell (type, trap = null, obstacle = null)
}
```

```
class Labyrinth
{
    Cell[,] Maze
    Labyrinth(width, height)
    {
        Maze = new Cell[height, width]
        MazeAlgorithm.GenerateMaze(this)
    }

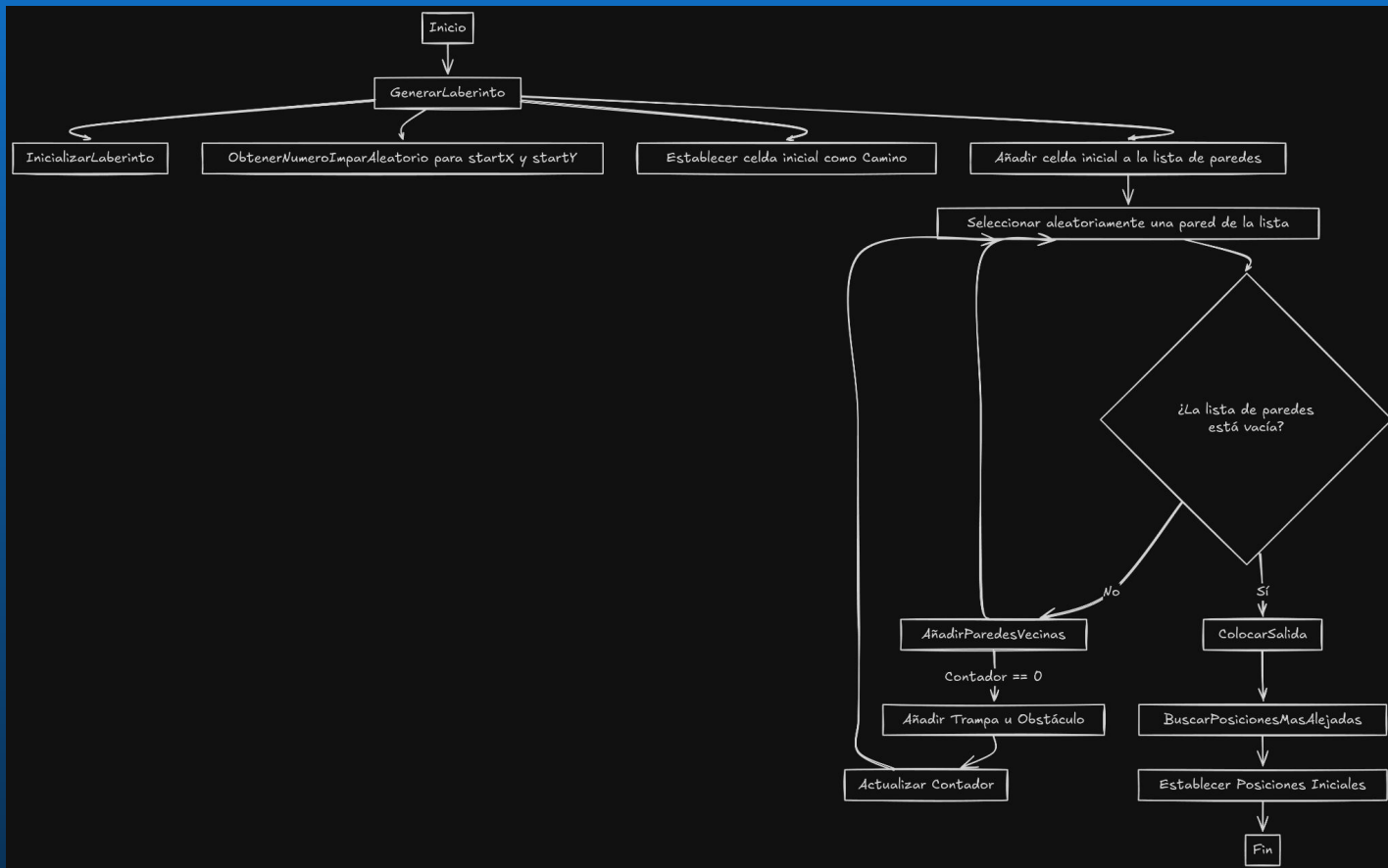
    public string GetCellType()
    {
        return Type switch
            CellType.Wall => "Wall",
            ...
    }

    void CheckAndActiveTrap(Player player)
    {
        cell = Maze[player.Position.RowPosition, ...]
        if (cell.GetCellType() == "Trap")
            cell.Trap.Active(player)
    }
}
```



# Flujo Principal

/Logic/MazeAlgorithm.cs



# Funciones

/Logic/MazeAlgorithm.cs

InicializarLaberinto

Para cada y en altura



Para cada x en ancho



Establecer celda como Pared

ObtenerNumeroImparAleatorio

Obtener numero aleatorio



¿Es  
par?

Si



Añadir 1 al numero

No



Mantener numero

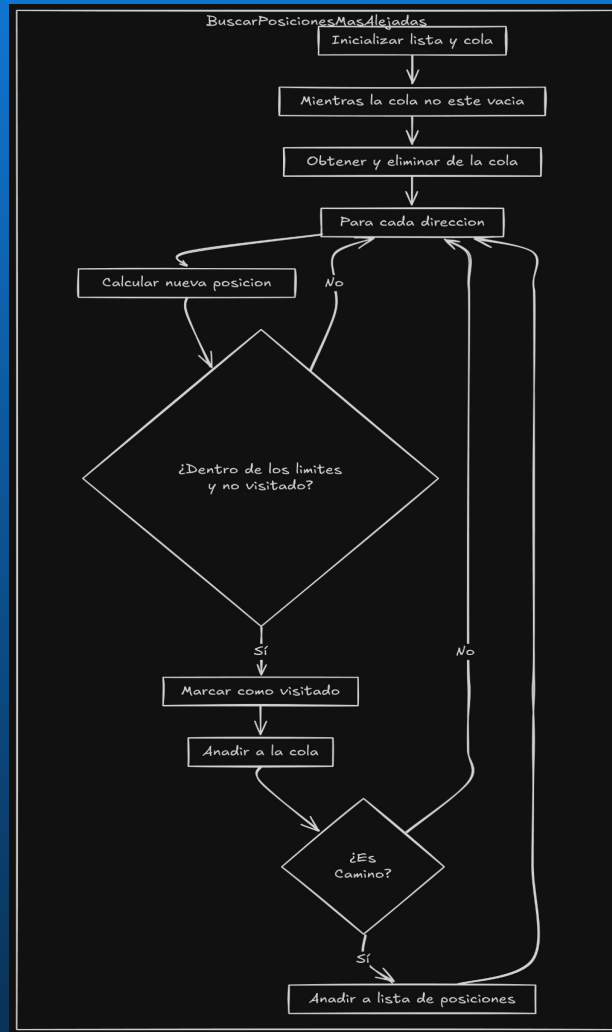
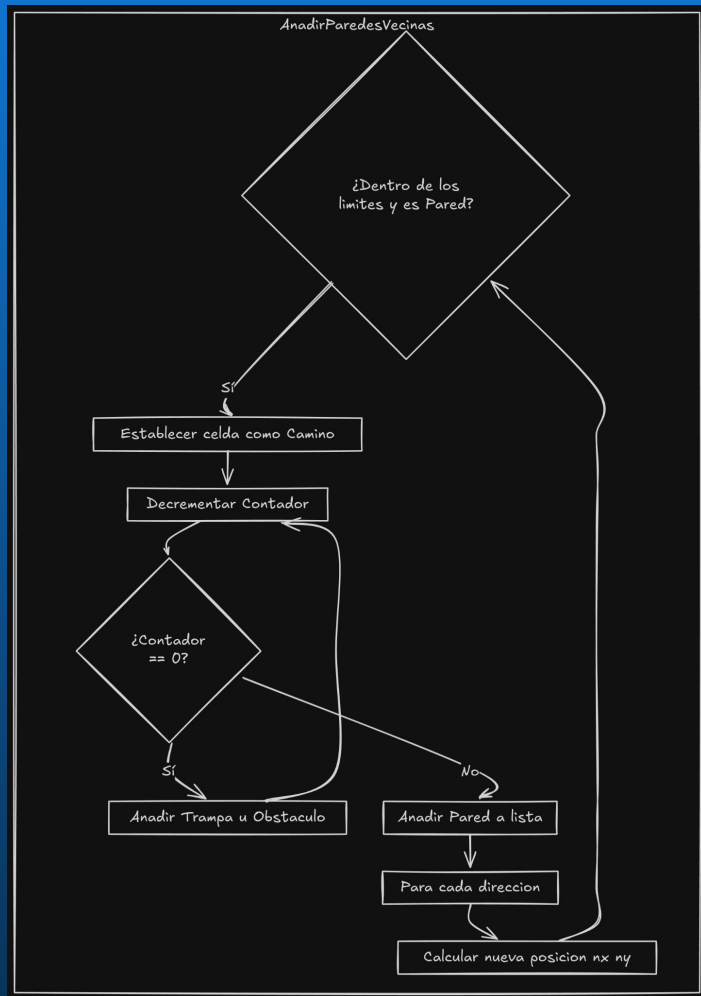
ColocarSalida

Seleccionar borde al azar



Colocar celda de salida en borde seleccionado





# Flujo del proyecto

## /Data/LabyrinthData.cs

```
class LabyrinthData
{
    SelectedSize = 15
    Labyrinth Game = new Labyrinth(SelectedSize, SelectedSize)

    void ResetLabyrinth()
    {
        Game = new Labyrinth(SelectedSize, SelectedSize)
        ValidationService.ResetValidationGame(Game)
    }
}
```



# Flujo del proyecto

## /Data/ObstacleData.cs

ObstacleData:

Variables:

- random = objeto Random
- ObstacleDefinitions = [  
    (100, "Rock", "/images/Obstacles/Rock.png")  
]

Function CreateRandomObstacle():

- obstacleDefinition = Seleccionar al azar un obstáculo de ObstacleDefinitions
- Crear y retornar un nuevo Obstacle con los atributos del obstacleDefinition

Function ResetObstacles(labyrinth):

Para cada celda en el laberinto:

- Llamar cell.Obstacle.Reset()





# /Data/TrapData.cs

TrapData:

Variables:

- Traps = [

Trampa de Veneno:

- Nombre: "Trampa de Veneno"
- Descripción: "Un gas venenoso del anime 'Naruto' que reduce la salud del jugador en 20 puntos."
- Imagen: "/images/Traps/Green Smoke.png"
- Acción: Reducir salud del jugador en 20 puntos

Trampa de Parálisis:

- Nombre: "Trampa de Parálisis"
- Descripción: "Electricidad del 'Raikiri' del anime 'Bleach' que inmoviliza al jugador por un turno."
- Imagen: "/images/Traps/Thunder.png"
- Acción:
  - Si el jugador y el héroe están seleccionados:
    - Añadir efecto temporal de parálisis al jugador
    - Movimientos restantes del TurnManager = 0
    - Pasar al siguiente turno en TurnManager

Trampa de Rc Cells:

- Nombre: "Trampa de Rc Cells"
- Descripción: "Una explosión de Rc Cells del anime 'Tokyo Ghoul' que reduce la salud del jugador en 15 puntos y su velocidad en 1 punto durante 3 turnos."
- Imagen: "/images/Traps/Red Smoke.png"
- Acción:
  - Reducir salud del jugador en 15 puntos
  - Si el héroe está seleccionado:
    - Añadir efecto temporal de Rc Cells al jugador
    - Reducir movimientos restantes del TurnManager en 1
  - Si movimientos restantes  $\leq 0$ :
    - Pasar al siguiente turno en TurnManager

]



# /Data/HeroData.cs

HeroData:

Variables:

- HeroCharacters = [

- Kaneki:

- Descripción: "Inicialmente un estudiante universitario normal..."
    - Nivel: 4
    - Salud: 70
    - Ataque: 35
    - Poder: Kagune Devorador
      - Descripción: "Despliega tentáculos afilados..."
      - Nivel: 3
      - Acción:
        - Daño en área 3x3 alrededor del jugador.
        - Roba vida de los enemigos dañados.
        - Recupera la mitad del daño infligido como salud del jugador.

Eren Yeager:

- Descripción: "Determinado a destruir a los titanes..."
- Nivel: 5
- Salud: 80
- Ataque: 38
- Poder: Titán Fundador
  - Descripción: "Transformación que destruye todas las paredes..."
  - Nivel: 4
  - Acción:
    - Destruye paredes adyacentes en un área 3x3.
    - Afecta a jugadores enemigos cercanos, dejándolos aturdidos durante 2 turnos.



## Goku:

- Descripción: "Un guerrero Saiyajin que busca constantemente desafíos..."
- Nivel: 5
- Salud: 100
- Ataque: 40
- Poder: Teletransporte
  - Descripción: "Teletransporta a Goku 6 casillas..."
  - Nivel: 4
  - Acción:
    - Teletransporta a Goku 6 casillas en la dirección que está mirando.
    - Aumenta temporalmente la velocidad del jugador por 2 turnos.



## Naruto Uzumaki:

- Descripción: "Marcado desde pequeño como un paria..."
- Nivel: 4
- Salud: 80
- Ataque: 32
- Poder: Modo Sabio
  - Descripción: "Aumenta todas las estadísticas..."
  - Nivel: 3
  - Acción:
    - Aumenta temporalmente el ataque y la velocidad del jugador por 4 turnos.



# /Data/HeroData.cs

## Monkey D. Luffy:

- Descripción: "Es un joven pirata con el sueño de encontrar el One Piece..."
- Nivel: 4
- Salud: 70
- Ataque: 34
- Poder: Gear Second
  - Descripción: "Aumenta velocidad y ataque, pero pierde salud"
  - Nivel: 3
  - Acción:
    - Aumenta temporalmente la velocidad y el ataque del jugador.
    - Reduce la salud del jugador en 10 puntos.
    - Si la salud llega a 0, restablece la posición y estadísticas del jugador.

## Saitama:

- Descripción: "Conocido como el héroe por diversión..."
- Nivel: 5
- Salud: 100
- Ataque: 100
- Poder: Puñetazo serio
  - Descripción: "Destruye TODO en lo que este en su camino"
  - Nivel: 5
  - Acción:
    - Destruye paredes, obstáculos y trampas en su camino.
    - Daña a jugadores enemigos en su camino.
    - Si la salud de un enemigo llega a 0, restablece su posición y estadísticas.

]



# Flujo del proyecto

## /Data/PlayerData.cs

PlayerData:

Variables:

- NumberOfPlayers = 1
- Players = []
- InitialPositions = []

Function ResetPlayerPositions():

Para cada jugador en Players:

- Establecer InitialPosition del jugador a la posición inicial correspondiente
- Establecer Position del jugador a la posición inicial correspondiente
- Llamar Players[i].ResetHasWon()

Function InitializePlayers(numberOfPlayers):

Si InitialPositions está vacío:

- Lanzar excepción "Debe generarse el laberinto primero"

- Establecer NumberOfPlayers a numberOfPlayers

- Limpiar lista de Players

- maxPlayers = Mínimo entre numberOfPlayers y el tamaño de InitialPositions

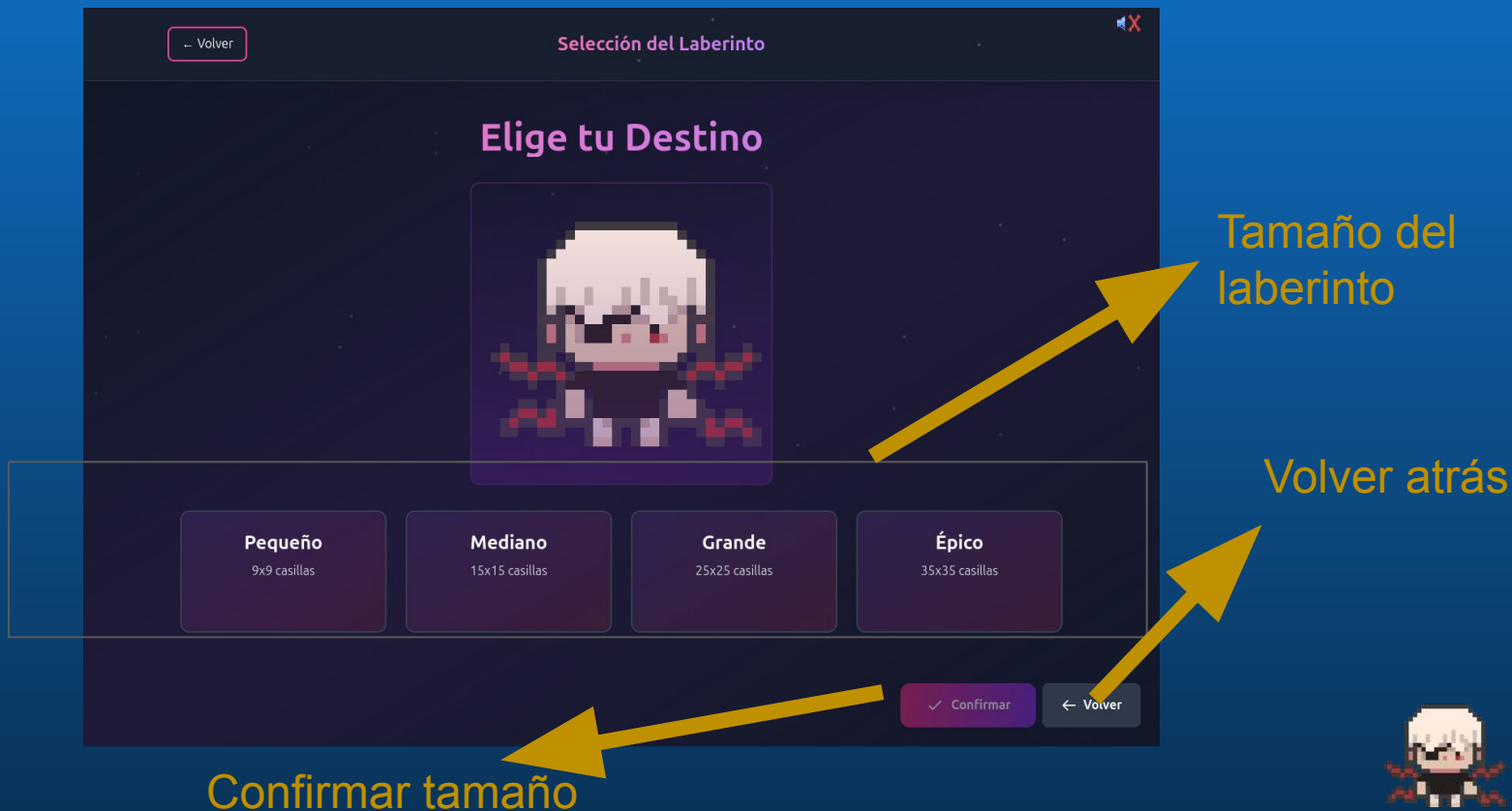
Para i desde 0 hasta maxPlayers:

- Obtener position de InitialPositions en el índice i
- Añadir un nuevo Player con position a la lista de Players

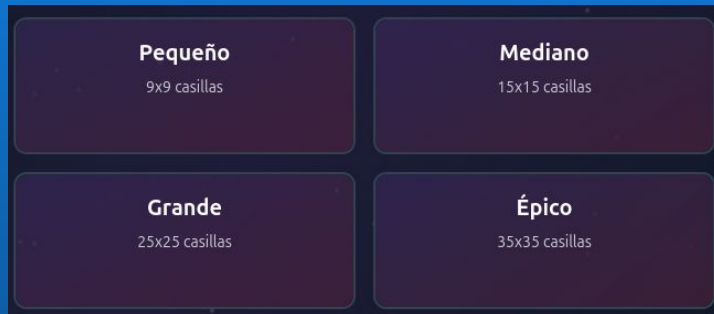


# Flujo del proyecto

## /MazeSizeSelection



# Flujo del proyecto



← Volver

Volver atrás

```
<button  
  @onclick="NavigateBack"(!)>
```

Tamaño del  
laberinto

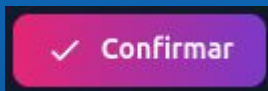
```
foreach(sizeOption in {9, 15, 25, 35}  
  <button @onclick="() =>  
    SelectSize(sizeOption)"
```

```
selectedSize = 0  
SelectSize(int size)  
  selectedSize = size
```



# Flujo del proyecto

Confirmar tamaño



```
void ConfirmSize()
{
    if (selectedSize > 0)
        LabyrinthData.SelectedSize = selectedSize
        Navigation.NavigateTo("/PlayerSelection")
}
```





# Flujo del proyecto

## /PlayerSelection

[... Volver](#)

Selección de Jugadores

Selección de Jugadores

Elige cuántos guerreros enfrentarán el laberinto

1 Jugador

Comenzar Aventura

Botón de  
Confirmar  
número de  
jugadores



# Flujo del proyecto

Botón de Confirmar número de jugadores

Comenzar Aventura

```
int numberOfPlayers = 1
void ConfirmNumberOfPlayers()
{
    PlayerData.InitializePlayers(numberOfPlayers)
    Navigation.Navigateto("/Characters")
}
```



Sección de Personajes




Volver

Selección de Guerreros

X

Jugador 1 - Elige tu personaje



Kaneki

4

70

35

Velocidad

Vida

Ataque

Poder Especial

Kagune Devorador    Coldown 3

Despliega tentáculos afilados que dañan en área y roban vida

Inicialmente un estudiante universitario normal, su vida cambia drásticamente cuando es transformado en mitad ghoul tras un accidente. Kaneki lucha por adaptarse a su nueva identidad y

Mostrar más



Eren Yeager

5

80

38

Velocidad

Vida

Ataque


Poder Especial

Títan Fundador    Coldown 4

Transformación que destruye todas las paredes adyacentes y aturde enemigos

Determinado a destruir a los titanes que amenazan a la humanidad, Eren descubre sus habilidades como titán cambiante y se convierte en una figura clave en la lucha por la supervivencia. Su determinación y

Mostrar más



Goku

5

100

40

Velocidad

Vida

Ataque


Poder Especial

Teletransporte    Coldown 4

Teletransporta a Goku 6 casillas hacia la dirección que esté mirando y aumento de velocidad

Un guerrero Saiyjin que busca constantemente desafíos y entrenamientos para convertirse en el ser más fuerte del universo. Su carácter amable y protector lo lleva a defender la

Mostrar más



Naruto Uzumaki

4

80

32

Velocidad

Vida

Ataque


Poder Especial

Modo Sabio    Coldown 3

Aumenta todas las estadísticas mediante el chakra natural

Markado desde pequeño como un paria por tener el zorro de nueve colas sellado dentro de él, Naruto sueña con convertirse en Hokage, el líder y protector de su aldea. Su perseverancia y espíritu indomable lo

Mostrar más



Monkey D. Luffy

4

70

34

Velocidad

Vida

Ataque

Poder Especial

Gear Second    Coldown 3

Aumenta velocidad y ataque, pero pierde salud

Es un joven pirata con el sueño de encontrar el One Piece y convertirse en el Rey de los Piratas. Luffy tiene la habilidad de estirarse como goma gracias a la fruta del diablo que comió, lo que lo convierte

Mostrar más



Saitama

5

100

100

Velocidad

Vida

Ataque

Poder Especial

Puñetazo serio    Coldown 5

Destruye TODO en lo que este en su camino

Conocido como el héroe por diversión, Saitama tiene una fuerza abrumadora que le permite derrotar a cualquier enemigo de un solo golpe. A pesar de su increíble poder, busca desesperadamente un

Mostrar más

Jugador 1 ha seleccionado:

Saitama

Comenzar Aventura



# Código del componente



## Variables:

- Lista de personajes: `characters = HeroData.HeroCharacters`
- Índice de descripción completa: `ShowFullDescription = null`
- Índice de héroe seleccionado: `SelectedHeroIndex = null`
- Héroe seleccionado: `SelectedHero = null`
- Número de jugadores: `numberOfPlayers = PlayerData.NumberOfPlayers`
- Jugador actual: `currentPlayer = 1`

## Método `OnInitialized()`:

- Para cada jugador en `PlayerData.Players`:
  - Establecer `HeroSelected` del jugador a `null`
  - Restablecer `SelectedHeroIndex` y `SelectedHero` a `null`

## Método `ToggleDescription(int index)`:

- Alternar la descripción completa:
  - Si `ShowFullDescription` es igual a `index`, establecer a `null`
  - Si no, establecer a `index`

## Método `SelectCharacter(int index)`:

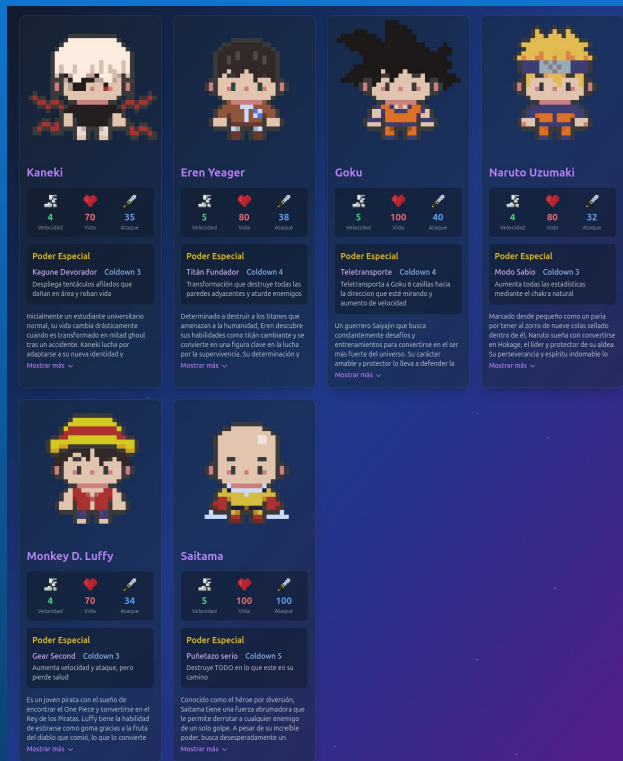
- Establecer `SelectedHeroIndex` a `index`
- Establecer `SelectedHero` al héroe en `HeroData.HeroCharacters` en el índice `index`
- Actualizar el estado (`StateHasChanged`)

## Método `PlayGame()`:

- Si `SelectedHero` no es `null`:
  - Establecer `HeroSelected` del jugador actual a `SelectedHero`
  - Establecer las estadísticas del jugador actual (`Health`, `Speed`, `Attack`) a las del héroe seleccionado
  - Incrementar `currentPlayer` en 1
  - Si `currentPlayer` es mayor que `numberOfPlayers`:
    - Navegar a la página del laberinto ("`/Maze`")
    - Si no:
      - Restablecer `SelectedHero` y `SelectedHeroIndex` a `null`

# Sección de Personajes

- Grid para mostrar personajes:
- Para cada personaje en la lista de personajes (characters):
  - Crear un HeroCard con el personaje actual
  - Establecer propiedades: Character, Index, OnSelect, OnToggleDescription, IsExpanded, IsSelected



# Tarjeta de Selección

Notificación de héroe seleccionado:

- Mostrar el héroe seleccionado (SelectedHero)
- Acción al iniciar el juego: PlayGame
- Jugador actual: currentPlayer



Jugador 1 ha seleccionado:

**Saitama**



Comenzar Aventura



Volver

Laberinto de Shikego

Reiniciar

Turno actual: Jugador 1

Movimientos: 4 (Velocidad 4)

70 HP

Habilidad:

Kagune Devorador

Cooldown

100 HP

Habilidad:

Puñetazo serio

Cooldown

Laberinto de Shikego

Cerrar

¡Victoria!

¡Jugador 1 ha ganado!

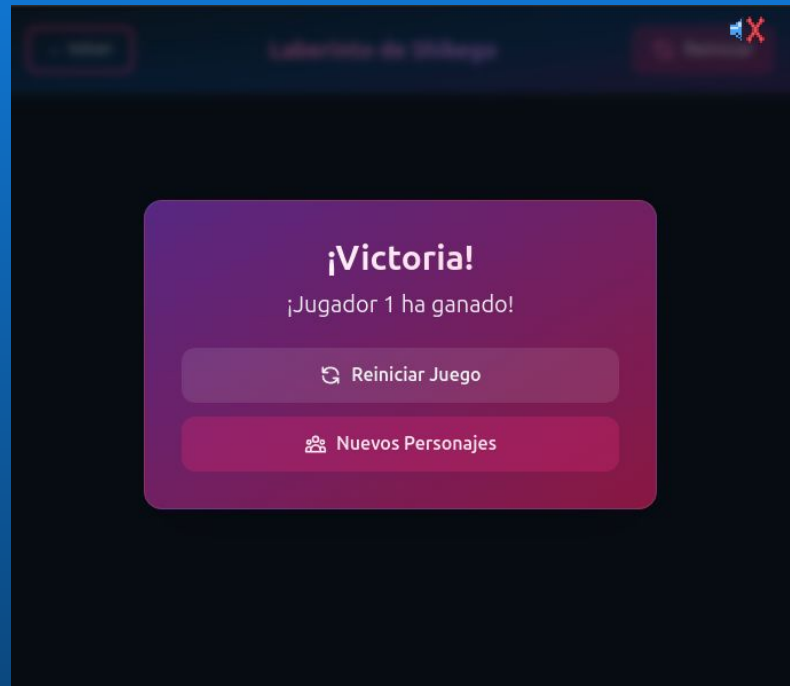
Reiniciar Juego

Nuevos Personajes



# Estado del juego

- Modal de juego ganado si showModal es verdadero
- Estado del juego si showModal es falso:
  - Turno actual
  - Movimientos restantes y velocidad del jugador actual





# Contenido Principal y Eventos

- Grid del laberinto con jugadores, trampas, obstáculos y meta
- Navegación del laberinto:
  - Mover jugador
  - Atacar
  - Usar poder
- Mostrar salud de los jugadores, su imagen, su habilidad y el cooldown

## Eventos:

- Script para escuchar teclas presionadas:
  - Agregar evento de escucha para teclas y llamar al método OnKeyPress cuando se presiona una tecla



# Métodos de Maze.razor

- OnAfterRenderAsync(firstRender):
  - Agregar evento de escucha de teclas en el primer renderizado
- MovePlayer(direction):
  - Mover al jugador en la dirección indicada
  - Verificar trampas y estado de victoria
- AttackPlayer():
  - Atacar al objetivo del jugador actual
- UsePowerPlayer():
  - Usar el poder del héroe seleccionado

- OnClickMaze(position):
  - Actualizar la dirección del jugador según la posición clicada
- OnKeyPress(key):
  - Realizar acción según la tecla presionada
- ResetMaze():
  - Reiniciar laberinto, turnos, posiciones de jugadores, obstáculos y validaciones del juego
- SelectNewCharacters():
  - Limpiar lista de jugadores, reiniciar turnos y obstáculos, y navegar a la página de selección de personajes





# Fin

# Gracias por su atención!

Link del proyecto: <https://github.com/joserafael0160/Maze-Runners>