# FASE DE ANÁLISIS

**Historia de Usuario HU01:**

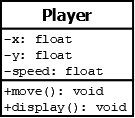
**Como** jugador **quiero** controlar una nave que pueda moverse, disparar y destruir asteroides **para** poder sobrevivir el mayor tiempo posible y obtener una puntuación alta.

**Tabla EPS (Entrada – Proceso – Salida):**

|  |  |  |
| --- | --- | --- |
| Entrada | Proceso | Salida |
| Teclas LEFT/RIGHT | Movimiento horizontal de la nave | Nave se mueve dentro de límites de pantalla |
| Tecla ESPACIO | Creación y disparo de proyectiles | Proyectiles se desplazan verticalmente |
| Tiempo de juego | Generación periódica de asteroides | Asteroides caen desde arriba |
| Colisión proyectil-asteroide | Detección de colisión y destrucción | Asteroides destruidos, puntuación aumenta |
| Colisión nave-asteroide | Detección de colisión y reducción de vidas | Vidas disminuyen, game over al llegar a 0 |
| Eventos de juego | Actualización de HUD en tiempo real | Muestra vidas y puntuación actual |

# FASE DE DISEÑO

**Diagrama de clases propuesto:**



# FASE DE CODIFICACIÓN

**Lista de Tareas HU01:**

Lista de Tareas HU01:

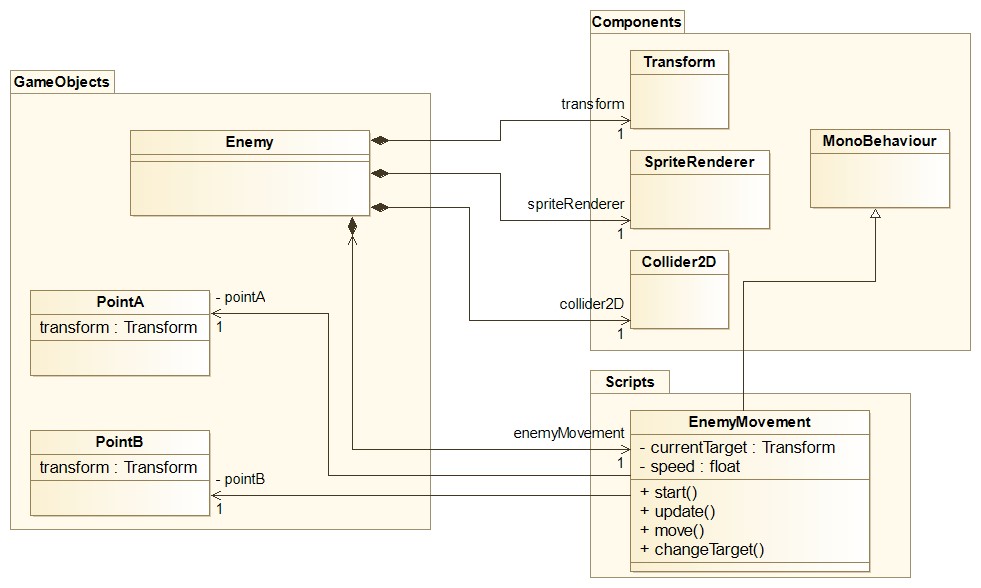
Objetivo Central: Implementar juego completo Nave vs Asteroides:

* Crear clase Player con movimiento y disparo
* Crear clase Bullet con movimiento vertical
* Crear clase Asteroid con movimiento vertical
* Crear clase HUD para mostrar información
* Implementar detección de colisiones bala-asteroide
* Implementar detección de colisiones nave-asteroide
* Implementar sistema de vidas y puntuación
* Configurar generación periódica de asteroides
* Probar juego completoProbar en escena.

**Criterios de Aceptación HU01:**

* La nave se mueve horizontalmente con las flechas LEFT/RIGHT
* La nave dispara proyectiles con la tecla ESPACIO
* Los asteroides aparecen periódicamente desde la parte superior
* Los proyectiles destruyen asteroides al colisionar
* Las colisiones nave-asteroide reducen las vidas
* El HUD muestra vidas y puntuación en tiempo real
* Game over cuando las vidas llegan a cero

**Diagrama de clases de implementación**



**Documentación del código Script EnemyMovement**

1 using UnityEngine;

2

1. /// <summary>
2. /// Permite que el gameObject patrulle entre dos puntos
3. /// </summary>
4. public class EnemyMovement : MonoBehaviour
5. {

## 8 #region Variables

1. /// <summary>
2. /// Posición a la que actualmente se dirige el game object
3. /// </summary>
4. private Transform currentTarget;
5. /// <summary>
6. /// Velocidad de desplazamiento
7. /// </summary>
8. private float speed;
9. /// <summary>
10. /// Representa el transform del punto A
11. /// </summary>
12. [SerializeField]private Transform pointA;
13. /// <summary>
14. /// Representa el transform del punto B
15. /// </summary>
16. [SerializeField]private Transform pointB;
17. #endregion

26

## 27 #region Unity life cycle

28

1. // Start is called once before the first execution of Update after the
2. MonoBehaviour is created
3. void Start()
4. {
5. currentTarget = pointA;
6. speed = 2f;
7. }

36

1. // Update is called once per frame
2. void Update()
3. {
4. Move();
5. }
6. #endregion

43

## 44 #region methods

1. /// <summary>
2. /// Actualiza la posición del enemigo hacia el objetivoActual
3. /// </summary>

## 48 public void Move()

1. {
2. // Cambio de posición
3. transform.position = Vector3.MoveTowards(
4. transform.position,
5. currentTarget.position,
6. speed \* Time.deltaTime);

55

1. // evaluar ejecutar cambio de objetivo
2. if(Vector3.Distance(transform.position,currentTarget.position) < 0.05f)
3. {
4. ChageTarget();
5. }

61

62 }

63

1. /// <summary>
2. /// Cambiar la posición objetivo
3. /// </summary>
4. public void ChageTarget()
5. {
6. if(currentTarget == pointA)
7. {
8. currentTarget = pointB;
9. }

## 73 else

1. {
2. currentTarget = pointA;
3. }
4. }

78

79 #endregion 80

}