ITCR

Ingeniería en Computadores

Algoritmos y Estructuras de Datos 2

Documentación Proyecto 2

Estudiantes:
Esteban Campos Abarca - 2022207705
Jafet Díaz Morales - 2023053249
Steven Pérez Aguilar - 2024118003

I Semestre 2025

Tabla de Contenidos

Introducción

En este documento se presenta la documentación de un juego estilo "Tower defense" llamado Genetic Kingdom. El objetivo de diseñar e implementar este juego es poder aplicar algoritmos genéticos y de pathfinding en el lenguaje de programación C++ además de diseñar la solución del problema mediante Programación Orientada a Objetos (OOP).

Breve Descripción del Problema

El proyecto consiste en implementar un juego estilo "Tower Defense" ambientado en la edad media en C++ para desktop. El juego genera oleadas de enemigos de distintas clases y categorías. El jugador se encarga de colocar torres en lugares predeterminados para evitar que los enemigos crucen el puente del castillo. Después de cada oleada, los enemigos evolucionan haciendo más difícil evitar que los enemigos crucen el puente.

Se tienen los siguientes requerimientos:

001 Generar un mapa cuadriculado de tamaño fijo donde se puedan colocar torres en cualquier cuadro que no bloquee el camino al puente. El mapa tiene un único punto de ingreso de los enemigos y está en el lado contrario al puente del castillo

002 Cada torre tiene los siguientes atributos: daño, velocidad, alcance, tiempo de regeneración del poder especial, tiempo de recarga de ataque. Las torres atacan a los enemigos cuando están en su alcance. Con cada muerte de enemigo, se considera su categoría y tipo para devolver cierta cantidad de oro al jugador. La cantidad de oro retornado debe ser calculado de forma justa y consistente.

003 Todas las torres tienen 3 upgrades. Cada upgrade aumenta el daño que pueden causar y cada torre tiene ataques especiales que ocurren cada cierto tiempo con una probabilidad definida por los estudiantes. Los upgrades tienen un costo en oro. Cada upgrade es más caro que la anterior.

004 Hay 3 tipos de torres:
• Arqueros: bajo costo, alto alcance, poco daño, tiempo de recarga de ataque bajo
• Magos: costo medio, alcance medio, daño medio, tiempo de recarga de ataque medio
• Artilleros: costo alto, alcance bajo, daño alto, tiempo de recarga de ataque alto

005 El jugador puede ir colocando las torres en cada lugar disponible. Al seleccionar un lugar disponible, puede escoger el tipo de torre que desea crear. Cada torre tiene un costo en oro.

006 Los enemigos aparecen por oleadas. Cada oleada es una generación que evoluciona. Los enemigos pueden ser los siguientes:
• Ogros: son el enemigo más básico. Son resistentes a los arqueros y débiles contra la magia y la artillería. Son lentos.
• Elfos Oscuros: son resistentes a la magia, pero débiles a los arqueros y a la artillería. Son muy rápidos
• Harpías: solo pueden ser atacadas por magia y arqueros. Tienen una velocidad intermedia
• Mercenarios: son débiles a la magia, pero resistentes a arqueros y artillería.
Cada enemigo tiene atributos como:
• Vida
• Velocidad
• Resistencia a flechas
• Resistencia a la magia
• Resistencia a la artillería

007 Cada generación selecciona los individuos con el mejor fitness, los cruza e ingresa los nuevos individuos a la población. Pueden ocurrir mutaciones con cierto grado de probabilidad. El estudiante debe definir cada elemento del algoritmo genético y explicarlo adecuadamente en la documentación. Las oleadas son de tamaño variable y se generan con un intervalo parametrizable.

008 Los enemigos utilizan Pathfinding A* para encontrar el camino hacia el puente del castillo.
20

009 El juego muestra un panel con estadísticas como:
• Generaciones transcurridas
• Enemigos muertos en cada oleada
• Fitness de cada individuo de la oleada
• Nivel de cada torre
• Probabilidad de mutaciones y cantidad de mutaciones ocurridas

Descripción de la solución

Se procederá a explicar la solución en referente a cada requerimiento.

Requerimiento 0:

o.

Requerimiento 1:

o.

Requerimiento 2:

o.

Requerimiento 3:

;

Requerimiento 4:

.

Requerimiento 5:

o.

Requerimiento 6:

o.

Requerimiento 7:

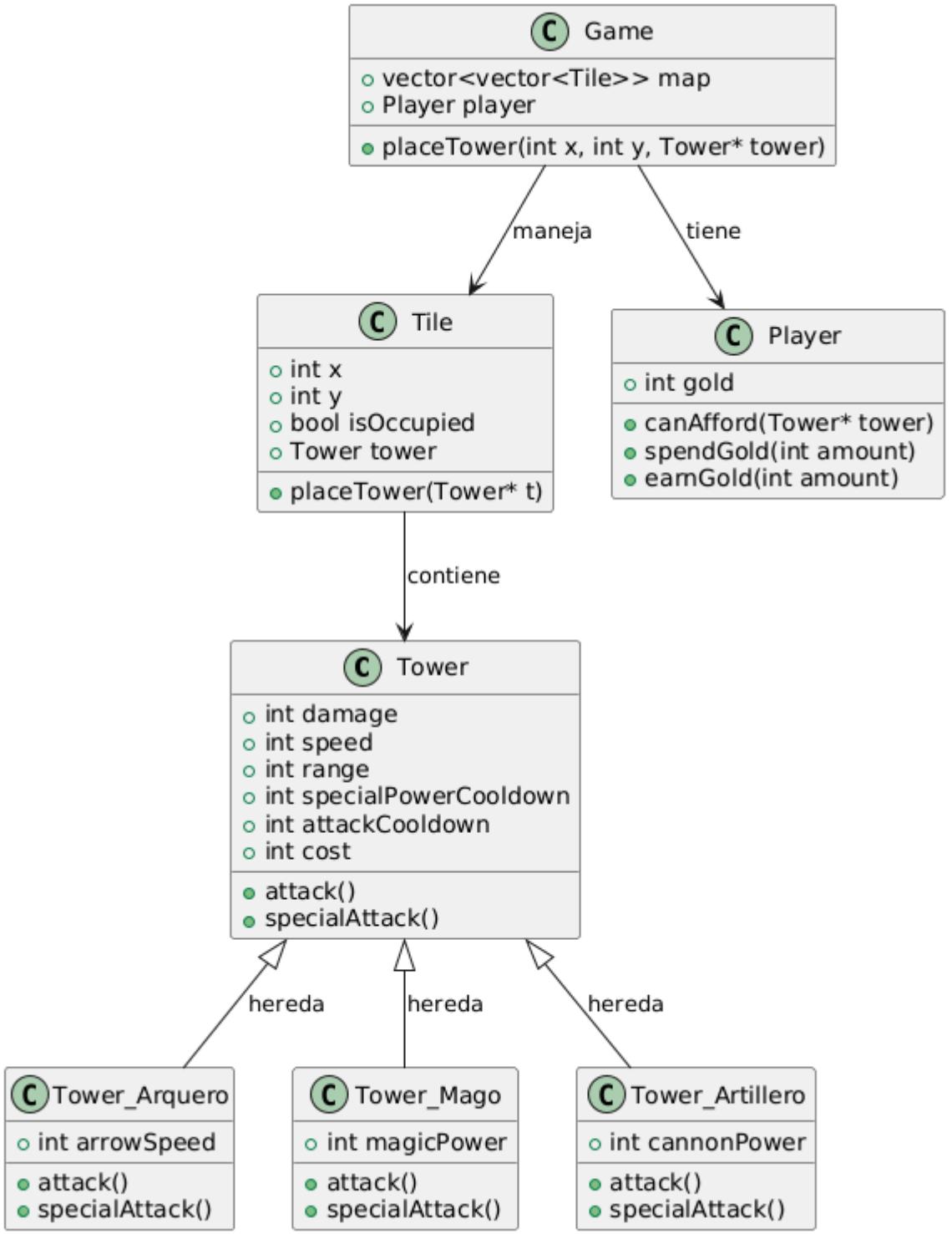
L.

Requerimiento 8:

Ss

Diagramas UML

A continuación se muestran los diagramas UML con las clases más importantes del proyecto.



Enlace a github:
https://github.com/EstebanACamposA/campos-diaz-perez-genetic-kingdom/tree/master