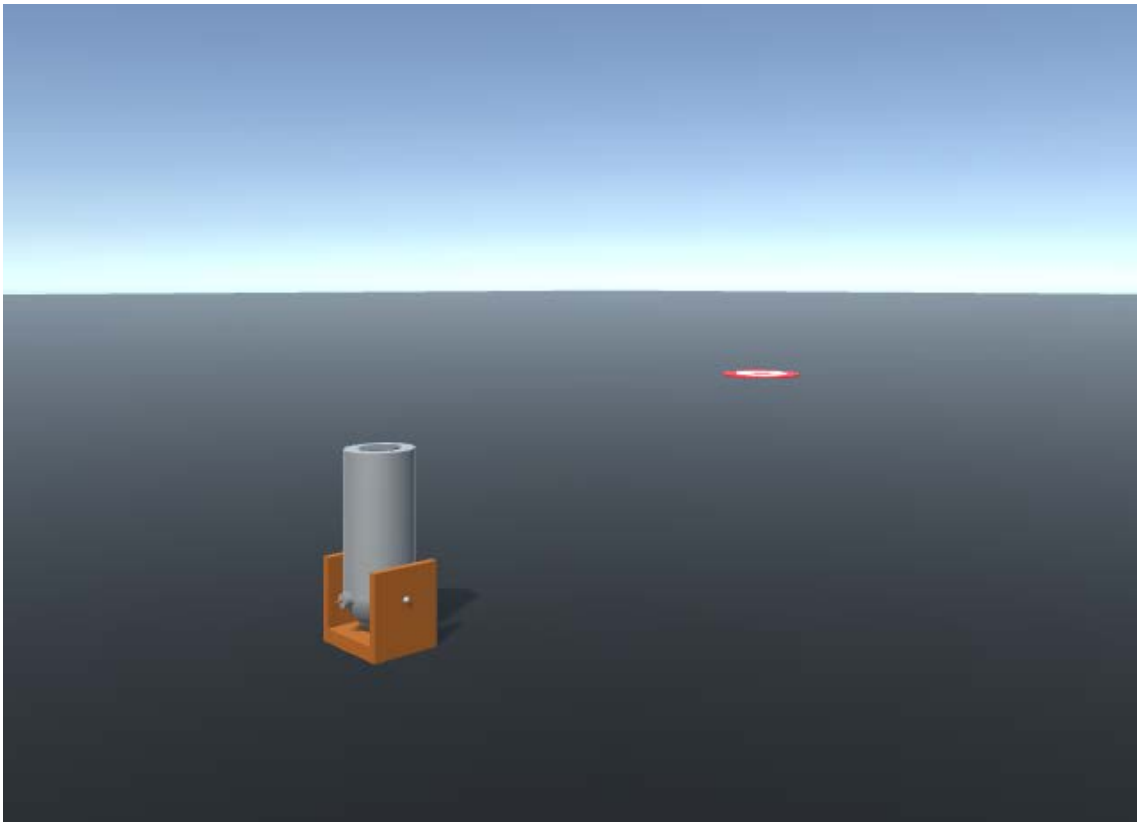
Hill Climbing Algorithm

by Alfredo Pérez Pastor



Sumario

En esta práctica se intenta recrear el algoritmo Hill Climbing con físicas propias de Unity, haciendo que calcule tridimensionalmente dos ángulos de rotación y el impulso del proyectil.

Introducción

Para implementar este algoritmo, de primeras se dispara una proyectil de forma aleatoria, y se recoge la mínima distancia con el objetivo, de esta forma se calculan las puntuaciones.

Si esta puntuación es mejor que las anteriores se dispara una cantidad de balas determinadas, con un margen establecido, de esta forma vamos escalando.

En caso de mejorar, volvemos a disparar las balas con los márgenes, hasta llegar al pico más próximo.

El disparo principal randomizado, será disparado un número determinado dado por el usuario.

Clases

Todo está englobado en una clase, que controla los proyectiles y sus puntuaciones:

```
// Dispara de forma aleatoria una bala con un margen.
void IARandomShotBall() ...

// Dispara y guarda los valores de los margenes generados
void ShotMarginValues(BallInfo ball) ...

// Devuelve si todos los jugadores han terminado
bool RefreshAllScores()...


bool RandomShotIsBest() ...
// Devuelve la distancia de una bala con el target
float GetScore(GameObject ball) ...

// Manejo Manual del cañon para pruebas
void KeyBoardController() ...

// Setea todos los BallInfo Y les añade el margen deseado en todas las dimensiones.
void FindMarginValues(BallInfo ball) ...

// Modifica un BallInfo con el margen establecido.
BallInfo ModifieBallInfo(BallInfo ball, int incrementBaseAngle, int incrementCanonAngle, int incrementImpulse) ...

// Dispara solo una bala, y guarda sus atributos en el BallInfo
void IAShotBall(float baseAngle, float canonAngle, float impulse, int ballListPosition) ...

// Rotacion de la base del cañon --> Horizontal
void RotateCanonBaseTo(float Angle) ...

// Totacion del cañon --> Vetical
void RotateCanonTo(float Angle)...

// Genera una pull con el numero suficiente, para ver todas las variaciones
void GenerateBallPull(int pullSize) ...

// Disparo fisico de una bala, con el impulso que se le indique
void ShotBall(GameObject ball, float impulse) ...

// Para la fuerza de una bala para que no repercuta al volver a dispararse
void stopForces(GameObject ball) ...

// Guia de direccion de las balas.
void ShowRayGuide() ...
```