# Programación Lúdica: Treasure Hunters

José Miguel Colella Carbonara
José Manuel Gómez

June 8, 2014

## 1 Introducción

En este documento queremos denotar los aspectos más relevantes en el desarrollo de nuestro juego *Treasure Hunters* usando como base la herramienta *pygame* que proporciona los mecanismos basicos para construir un videojuego usando **python**.

Las secciones que vamos a destacar son las opiniones sobre *pygame* como herramienta para el desarrollo de un videojuego, el diseño del juego en base a como se ha estructurado, los mecanismo de inteligencia artificial, cosas particulares del juego, y nuestras conclusiones sobre el camino que hemos recorrido desde el diseño hasta la finalidad de tener un juego funcional.

## 2 Pygame

*Pygame* proporciona un fundación sólida para el diseño de un videojuego y los objectos que estarán presentes. En base a colisiones, creación de menus, gestion de imagenes, y sprites pygame facilita la construcción de objetos para la creación de un videojuego. Una cosa de notar es que por ejemplo, mecanismos de guardar, cargar, estructuración de videojuego, y la inteligencia artificial la tiene que crear el desarrollador desde nada.

## 3 Diseño del Juego

Nuestro diseño para este juego ha utilizado el paradigma de programación dirigida a objectos. Hemos usado las ventajas de la herencia para crear objetos escalables, faciles de usar y extender.

Para los personajes que forman parte del juego que incluye el *MainCharacter*, *Enemy*, y *Robot*, todos se extienden de una clase llamada *Character* que proporciona los mecanismos de movimiento, disparo, interacción con otros objetos en el juego, etc...

# 4 Inteligencia Artificial

Para la construcción de la Inteligencía Artificial se ha pensado en una construcción de IA multinivel, en el cual tenemos un nivel que tiene acciones de alto nivel:

- change_zone(asset, new zone)
- change_gun(asset)
- pick_up(asset, object)
- drop(asset)
- hurt(asset, asset)
- switch(asset, lever)

mientras que el otro nivel se encarga de las acciones de bajo nivel:

- fire(direction)
- move(direction)
- ...

Para hacer esto se ha separado las responsabilidades en tres clases:

1. Agent
2. AgentServer
3. Think

La idea es tener a AgentServer construye objetos Think y Agent interconectados y mantiene una copia del estado del juego lista para ser procesada por los mecanismos de IA. Se ha usado el paradigma de programación concurrente para poder separar los procesos de la IA de la hebra principal.

# 5 Notas Adicionales

Unas de las cosas que quisieramos destacar es el sistema de guardar y cargar y el potencial que ofrece para una futura integración multijugador red. Se ha usado el sistema de intercambio ligero, *JSON*, en el cual se guarda el estado actual del juego, almacenando las cosas más importantes; coordenadas, información sobre el objeto a coger, sobre los enemigos, etc...
Para un sistema multijugador se transferiría dicha información en formato *JSON*, a los otros jugadores

# 6 Conclusión

## 6.1 José Colella

Por mi parte, pienso que para haber tenido que construir la estructura del juego, diseñando un modelo escalable, y creando la Inteligencia Artificial desde nada, hemos creado un juego con gran potencial, y que en su estado actual ofrece un opción completa.
Desde tener un menu, uno para acceder al juego ó cargar un estado previo y otro que se usa para acceder desde el juego y guardar el estado, crear música para menu y videojuego, se ha creado un producto lo más completo posible.
Estoy muy satisfecho con el producto que hemos sacado a luz con mi compañero

## 6.2 José Manuel Gomez