Simulación en UE4 para realizar Aprendizaje Automático

Jon Irigoyen Cañas

Resumen—Si se le preguntara a cualquier entendido qué avances tecnológicos darán forma a nuestro futuro, la mayoría por no decir todos, mencionarían los coches autónomos. Pero para que esta tecnología se popularice se debe minimizar al máximo el margen de error. Aquí es donde entran los algorismos de machine learning, que consisten en entrenar software usando los datos apropiados para obtener el resultado esperado. Estos algoritmos requieren un proceso de aprendizaje, los cuales requieren a su vez entornos simulados. En este proyecto se creará todo el framework que se necesita para poder simular un algorismo machine learning aplicado a la conducción autónoma.

Palabras clave—Coches autónomos, aprendizaje computacional, Unreal Engine, simulación, sockets, video streaming, big data.

Abstract—If we ask any professional working in the computer research fields what kinds of technological advancements will shape our future, most of them, if not all of them, would probably mention self-driving cars. But to launch this technology to the mainstream level we'll need to minimize the possibility of error. Here's where computer learning comes into play helping the software learn the appropiate behaviours from selected data. These algorithms require a learning phase, which can be done in simulated environments to prevent cost and accidents. This project implements the required framework one would need to simulate a machine-learning algorithm designed towards self-driving cars.

Index Terms—Autonomous cars, Machine Learning, Unreal Engine, simulation, sockets, video streaming, big data.

1 Introducción

Los coches autónomos están siendo una de las nuevas tecnologías más sobresalientes en los últimos años. Uno de los métodos para que un autómovil sea capaz de conducir sin ayuda humana es el uso del reinforcement learning. Este consiste básicamente en que, a partir de una serie de imágenes e información recogida del entorno, el automóvil utilice una red neuronal para identificar qué imágenes se corresponden con cada situación específica y reaccione actuando en consecuencia.

Así pues, una vez entrenado en un entorno determinado, el coche debería ser capaz de reaccionar correctamente frente a las futuras dificultades que se le planteen. No obstante, para asegurarse de que el aprendizaje sea seguro y relativamente rápido, existe la posibilidad de usar entornos simulados.

El objetivo de este TFG es la creación de un entorno simulado en el que aplicar *reinforcement learning* para el aprendizaje de un coche autónomo.

El proyecto consiste en la creación de un entorno 3D utilizando la herramienta *unreal engine*, la cual se utiliza principalmente para el desarrollo de videojuegos, pero que es muy útil para el diseño de cualquier aplicación donde el aspecto gráfico tenga una especial importancia.

Unreal engine es una herramienta construida en C++ , que será el lenguaje utilizado para realizar la mayor parte del proyecto, aunque se trabajará mayoritariamente mediante la utilización de las librerías de *unreal* preexistenteses.

En el caso concreto de este proyecto, interesa que se pueda representar el prototipo de una ciudad o sección de la misma, para el entrenamiento de un coche simulado que usará *reinforcement learning*. Esto implicará el desarrollo de un sistema urbano que emule, lo más fielmente posible, a la realidad, con sistemas de regulación del tráfico y posibles atascos ocasionales.

No obstante, dada la corta duración del período de tiempo destinado a la realización del TFG, la complejidad urbana se verá limitada con respecto a la de una ciudad real, pero se ha procurado que se asemeje lo suficiente a la complejidad de una situación de conducción real, como para experimentar ampliamente con el sistema de *reinforcement learning*.

Cabe recalcar que el aprendizaje automático no forma parte del prototipo a elaborar, pero este debe permitir la fácil incorporación de una posible IA.

Las partes esenciales del proyecto serán:

- Un entorno 3D creado en *Unreal engine* que imite una situación de tráfico equiparable a las que nos encontramos en la vida real.

- Un sistema de *sockets* que permita al entorno comunicarse con el cliente donde se realizara el aprendizaje automático.
- Un método/protocolo con el cual estos *sockets* puedan enviar imágenes.
- Un coche especial dentro del entorno 3D que contenga una cámara ligada a este *socket*.(Fig.1)

Una vez dispongamos de este esqueleto, tendremos un entorno 3D preparado para realizar aprendizaje automático.

Figura 1. Cámara coche Machine Learning 1

2 Estado del arte

El uso de entornos 3D en el entrenamiento del software para vehículos autónomos es relativamente nuevo, pero no es exclusivo de este proyecto.

2.1 OpenAI

Existen iniciativas que usan videojuegos como punto de partida. La compañía OpenAi en particular ha lanzado la plataforma Universe para medir y entrenar algoritmos de inteligencia artificial (IA) en diferentes videojuegos. A partir de esta pueden usar el videojuego Grand Theft Auto 5 para entrenar coches autónomos.

Los resultados dependerán de la fidelidad que la simulación tenga respecto al mundo real. El juego que se use de referencia deberá ser lo más cercano a la realidad posible, de no ser así la IA no se podrá trasladar al mundo real. Por suerte vivimos en una época en la que algunos videojuegos alcanzan un gran nivel de fidelidad, debido al alto presupuesto del que disponen.

2.2 UETorch

Hay iniciativas no tan específicas como la anterior, pero que siguen el mismo principio, como por ejemplo UETorch de Facebook. UETorch es un open source que se integra en Unreal Engine 4 y permite extraer datos para usar en aprendizaje automático, tal y como se ha hecho en este proyecto.

3 Metodología

PARA realizar este trabajo ha habido que familiarizarse con la herramienta unreal engine (Fig. 2) y aprender de ella de forma constante. Se han utilizado para ello desde los tutoriales más generales y simples [1], hasta la documentación oficial [2] y los foros de ayuda [3], ampliando las consultas a medida que se ha ido incrementando la complejidad de los conocimientos a adquirir.

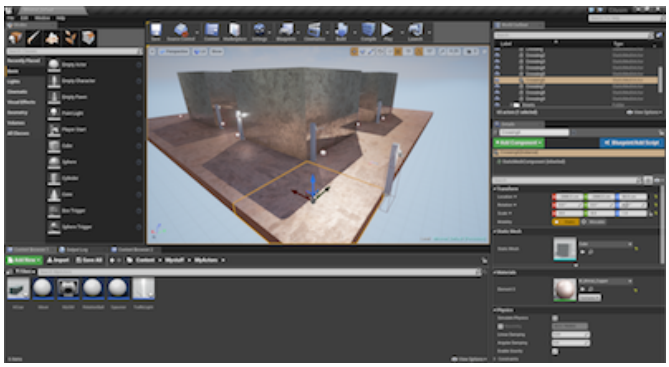


Figura 2. Unreal Engine

También ha sido útil expandir el conocimiento sobre aprendizaje automático, tanto a través de las fuentes proporcionadas por el director del proyecto [4,5] como mediante el curso online realizado paralelamente este semestre [6], el cual forma parte de la asignatura "Aprendizaje Computacional" de la UAB.

Por último, se consultaron fuentes que no estaban previstas en un principio, pero que se hicieron necesarias según iba avanzando el proyecto, como por ejemplo cuando hubo que transformar un sistema de coordenadas en otro [7].

La metodología para realizar el proyecto se ha inspirado en la propia de los procesos que siguen la metodología Scrum. Se han hecho esprints de una semana de duración que finalizaban con una reunión con el director de proyecto. En dichas reuniones se analizó lo realizado hasta la fecha y se tomaron decisiones sobre la dirección a seguir y sobre las prioridades a tener en cuenta en el siguiente esprint.

La programación se ha estructurado de forma que todos los actores implicados en la ciudad simulada heredasen de una clase general y no hubiesen dependencias entre ellos. De esa forma se podría en un futuro, si se desease, añadir nuevos tipos de actores sin aumentar la complejidad de forma exponencial.

Por otro lado, la lógica interna de los actores y sus gráficos estarán separados de forma que la primera se implemente en C++, mientras que los segundos se implementen en *unreal blueprints* (Fig. 3) que heredarán de las clases lógicas creadas en C++. De esta forma todo el diseño es modular y cambiar una pieza, aunque sea antigua, no implica reorganizar todo el sistema.

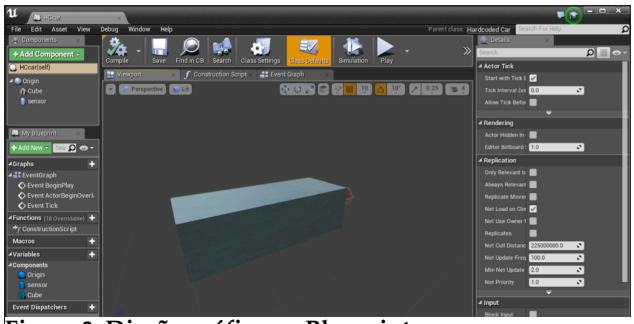


Figura 3. Diseño gráfico en Blueprint

4 DESARROLLO

4.1 Captura de requisitos

Requisitos funcionales:

1. Control remoto de un vehículo.
2. Retransmisión de imágenes del servidor al cliente.

Requisitos no funcionales:

1. Desarrollar el entorno 3D con la herramienta *unreal engine*.
2. Comunicación cliente/servidor mediante sockets que permitan el acoplamiento de una IA.
3. Simulación de trafico equiparable a la vida real dentro del entorno 3D.
4. Vehículos no controlables que sigan las normas de tráfico dentro de la simulación.

4.2 Diseño

4.2.1 Diseño del entorno 3D

El desarrollo del entorno 3D ha consistido en la representación de cuatro manzanas de edificios, cruzadas por dos calles y bordeadas con una circunvalación.

En todas las intersecciones hay semáforos que organizan el tráfico y todas las intersecciones, ofrecen más de una opción de giro, obligando a los coches a tomar la decisión de hacia dónde girar al llegar a cada semáforo. Los coches tienen todos las mismas dimensiones, van entrando por una sola calle y permanecen en el circuito durante todo el proceso (Fig. 4).

Para que estos coches sean capaces de tomar decisiones correctas se han creado tres tipos de actores:

- Semáforos: En el circuito hay un total de 17 semáforos que prohíben el paso si se encuentran en fase roja o amarilla y lo permiten si están en fase verde.
- Rotadores: En el circuito hay 16 rotadores que provocan un giro de X grados al coche que colisione con ellos, siendo X un valor que se determina desde el editor de Unreal.
- Movedores: En el circuito hay 16 movedores, que atraen a sus coordenadas al coche que colisione con ellos. Su función es corregir cualquier desfase provocado por los rotadores. En un mapa correctamente diseñado, el efecto causado por los movedores debería ser imperceptible.

Los coches están programados para no detener la marcha, excepto si se detectan entre ellos u otro actor del mapa les impide seguir hacia delante, pudiéndose provocar situaciones de atasco circulatorio, que serán cada vez más frecuentes a medida que aumente la densidad del tráfico.

Por último, para generar los coches se ha creado una clase "spawner" que genera X coches cada Y tiempo, siendo X e Y valores que se determinan desde el editor de Unreal.

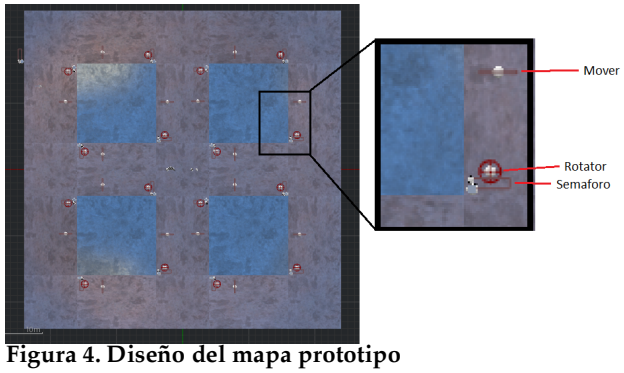


Figura 4. Diseño del mapa prototipo

4.2.2 Desarrollo de los sockets

Por otro lado tenemos los sockets, cuyo objetivo consiste en comunicar la aplicación de Unreal con una aplicación externa que tratará las imágenes recibidas por su socket y devolverá una instrucción de control para que el coche la ejecute. Toda la IA al otro lado del socket no se implementa en este proyecto, motivo por el cual, para nosotros será una "caja negra" que generará unos outputs para controlar el vehículo.

Los sockets se han desarrollado y testeado en windows, para lo cual están basados en los ejemplos que provee Microsoft [8].

Servidor:

El software desarrollado en unreal se conectará con el cliente y le enviará la imagen que el programa muestra por pantalla en tiempo real, a la vez que un conjunto de datos relevantes que, en estos momentos, son los siguientes:

- Estado del coche. ¿Se está cometiendo una infracción? Se usa un byte en vez de un bit en el caso

de que se quieran ampliar los códigos de error.

- Objeto con el que se ha cometido la interacción que ha generado la infracción. Esta información no es solo útil en el caso de un choque, también proporciona datos sobre otro tipo de infracciones. Ejemplo: semáforo, carril contrario, etc.
- Velocidad relativa con el objeto del punto anterior. Esta información sí es más concreta que las indicadas anteriormente, aunque en general un exceso de velocidad siempre puede influir sobre la gravedad de cualquier infracción.

El servidor recibirá, como respuesta a todo lo anterior, una orden de movimiento que aplicará al coche machine learning.

Cliente:

El cliente recibirá las imágenes y las generará en una pantalla en tiempo real. Este video en streaming funciona a una velocidad media de treinta fotogramas por segundo.

El cliente también reconocerá inputs de teclado y los enviará al servidor en tiempo real, siempre y cuando se mantenga la conexión. (Fig. 5).

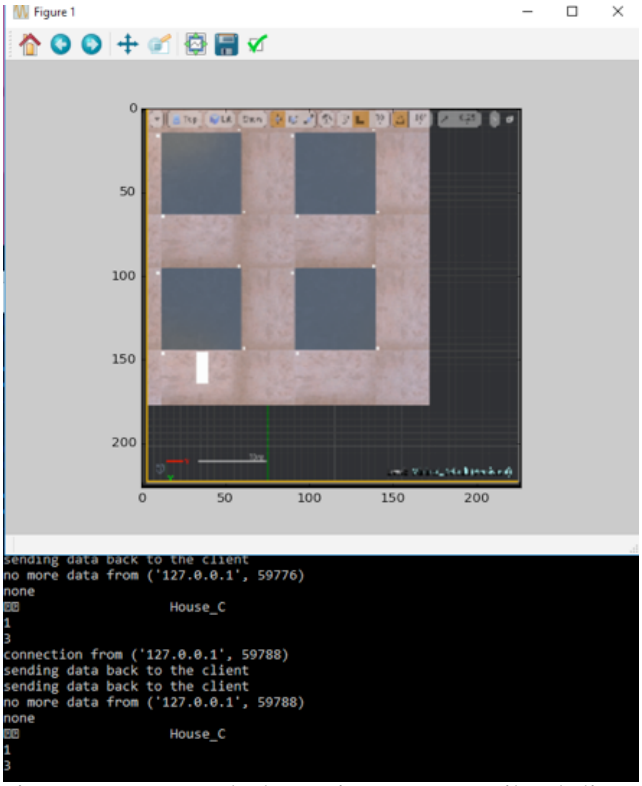


Figura 5. Estructura de datos e imagen que recibe el cliente

4.2.3 Captura y tratamiento de imágenes

Ha sido necesario implementar una cámara en el coche machine learning (ML), ya que su punto de vista es el que interesa al realizar aprendizaje a partir de imágenes. Dicha cámara se ha implementado en Unreal sin necesidad de escribir código C++, de forma que si el coche ML existe en el nivel actual, la cámara cambia para enfocar lo que el coche ML está viendo.

Para implementar esta funcionalidad, por un lado tenemos la clase IncidentManager en unreal que captura la imagen y la redimensiona si es necesario al tamaño 227x227 píxeles. (Fig. 6). Cada píxel se almacena como tres bytes, representando valores RGB en una matriz 227x227x3 y esta matriz se guarda en un buffer como una cadena de bytes, la cual será enviada finalmente por el socket.

Estos requisitos se han proyectado a partir del trabajo con IAs ya realizado en el cvc.

Figura 6. Ejemplo de *image scaling*

4.2.4 Tratamiento de incidencias de tráfico

Para generar los datos de las infracciones de tráfico se ha creado la clase IncidentManager, que no hereda de ninguna clase de unreal e implementa los métodos relacionados con generar y extraer dichos datos.

Para seguir una filosofía de diseño que organizase el código de forma apropiada, se incluyó todo el tratamiento de imágenes por parte del lado unreal en esta clase, que contiene el método GetImage(), que obtiene la imagen actual de una ventana, normalmente la de unreal, pero se

puede modificar fácilmente, si es necesario, en el *debugging*.

Así mismo, hay que tener en cuenta que el programa enviará imágenes en todo momento, no solo cuando haya una infracción. Para esos casos existen los métodos GetInfraction() y NoAccident(), que modifican las variables de la estructura, acordes con si ha habido una infracción o no.

La clase incluye también métodos de serialización, ya que los datos se enviarán todos por el socket en forma de array de bytes.
El método GetIncident() devolverá un array a partir de los otros métodos ya mencionados.

4.2.5 Control del coche aprendizaje automático

Por otra parte, estará el control del coche por parte del cliente, el cual ha sido implementado alterando el método principal de la clase Socket para que devuelva un char array con la respuesta del cliente.

El coche machine learning, que es el que llama a la función del socket para comunicarse, guarda la respuesta y la interpreta según el protocolo establecido de la siguiente forma:

"a" = Acelerar
"s" = Decelerar
"d" = Giro hacia la derecha.
"a" = Giro hacia la izquierda.

Los giros son de aproximadamente 3 grados, para otorgar así más libertad al cliente, pudiendo girar con el ángulo deseado si se mantiene apretada la tecla adecuada. Desde el otro lado, el script python que usará el cliente simplemente captura que tecla está siendo pulsada en ese ordenador y la envía como respuesta por el socket, es necesario que ambos lados sepan qué comandos se usan para cada orden.

4.3 Código

Diseño de clases (ver Apéndice 1)

C++

Clase	Líneas de código
SocketClient	196
MLcar	172
HardcodedCar	155
IncidentManager	118
TrafficLight	51
AbstractCar	46
CityActor	42
Spawner	35
Intersection	30
ForceRotation	23
Mover	17
Building	11

Python
Socket+image ploting script = 65 líneas de código

Se ha intentado respetar la filosofía de diseño de no crear clases de mas de 200 líneas de código o métodos de más de 20 líneas.

Como se puede ver en la lista, la mayor parte del código se ha centrado en la creación de sockets y los coches, tanto los automáticos como el controlado por el usuario.
Les sigue la clase IncidentManager dedicada a la administración de accidentes y la generación de imágenes. Y para acabar el resto de clases, de menor complejidad.

Por otro lado el código python es notablemente más breve, incluyendo el socket y el tratamiento de imágenes en tan solo 65 líneas.

4.4 Testing

Para asegurarse de que las diferentes partes del proyecto funcionan correctamente, se han usado diferentes métodos de testeo. Primero se utilizaron pruebas de exploración, para asegurarse de que los diferentes parametros modificables de la aplicación Unreal tuviesen unos valores que dieran un resultado apropiado. Más tarde se diseñó una pequeña aplicación, independiente del proyecto, para testear los sockets.

4.4.1 Pruebas de exploración Unreal

En unreal engine se han hecho testeos de exploración para aproximar dos valores, por un lado, la cantidad máxima de coches y por otro la frecuencia de entrada que genere un tráfico razonable. (Fig. 7).

Figura 7. Tráfico deseado tras la exploración

Haciendo esto se ha comprobado que en caso de sobrepoblación de coches sucede lo esperado, un atasco sin solapamientos.

4.4.2 Sockets

Para el desarrollo de los sockets se crearon proyectos cliente/servidor destinados a testear la funcionalidad de los mismos. El test consistió en abrir un fichero, enviarlo a través de un socket, salvar el fichero una vez recibido por parte del servidor y comparar ambos archivos para comprobar que son idénticos. Así mismo, se creo otro proyecto independiente para la captura y el tratamiento de las imágenes.

Finalmente, se creó un proyecto destinado a la recreación de una imagen a partir de un array de bytes, que se recibían a través de un socket. El único propósito de dicho proyecto era comprobar la funcionalidad de ambos sistemas. (Fig. 8).

Para testear estas funcionalidades se ha creado un servidor mockup con un pequeño algoritmo que recibe el mensaje y reconstruye la imagen, para lo cual se ha usado la librería GDI+[9]

Una vez realizadas las comprobaciones que demostraron el correcto funcionamiento de los distintos elementos, se procedió a implementarlos en la clase SocketClient del proyecto Unreal.

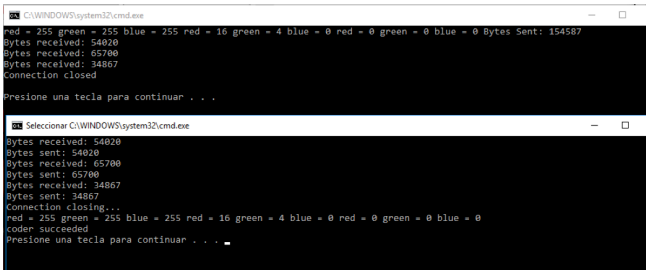


Figura 8. Testeo de sockets

4.4.3 Plotting y testing final

Finalmente, cabe mencionar que la funcionalidad de *video streaming* en el script de python no sería necesaria una vez implementada una AI, así como el hecho de que se muestren los datos por pantalla. Estas funcionalidades están implementadas para asegurarse de que el programa hace lo que tiene que hacer y poder visualizar fácilmente todos los módulos funcionando en conjunto (Fig. 9).

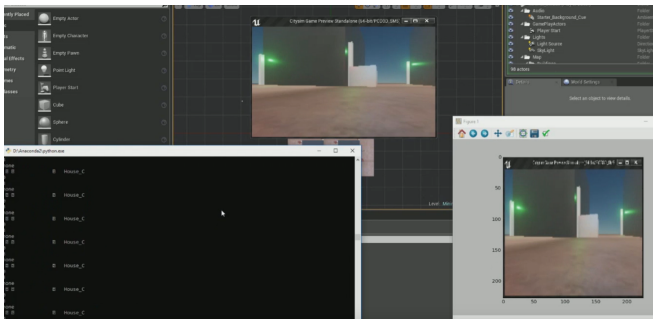


Figura 9. Vista final del proyecto

5 GUÍA DE USUARIO

Los pasos para utilizar la versión final de la aplicación son los siguientes:

1. Ejecutar la simulación Unreal.
2. Ejecutar el script Python.
3. Pulsar las teclas “a-w-s-d” de movimiento del coche dentro de la ventana de control de python.

Así pues, es necesario poder ejecutar python scripts (distribución Anaconda) y utilizar un sistema operativo Windows.

6 PROBLEMAS QUE SE PRESENTARON DURANTE EL DESARROLLO

6.1 Choque y rotación de los coches

Se detectó un problema en los coches no controlados de la simulación. Para que las colisiones fueran apropiadas y ningún objeto se solapara hubo que mover el origen de los coches al morro de los mismos.

Esto provocó que Unreal reconociera el eje de rotación como el morro, ya que asoció el eje de rotación al origen, hecho que generó un efecto derrape cuando los coches giraban.

Este problema no es algo que parezca difícil de solucionar, ya que se calculó que, en el peor de los casos, bastaría con modificar la rotación usando algebra y, en el mejor, indicarle a Unreal que desasocie el eje de rotación del origen del objeto. Por desgracia hubo que empezar el desarrollo de sockets y tratamiento de imágenes, lo que comportó que, al darle prioridad a estos temas, el problema pasase a un segundo plano y persista en la actualidad.

6.2 Tratamiento de imágenes

La mayor fuente de problemas en este campo vino determinada por la dificultad de la obtención de la imagen del programa Unreal. En un principio, se intentó trabajar con las librerías Unreal para obtener la imagen de la aplicación usando la función FscreenshotRequest [10], pero dicha función guarda la imagen constantemente en memoria y eso impedía que fuese óptima.

Por otro lado, se investigó la opción de que la aplicación Unreal se representase a 227x227 y luego se escalara a la resolución que el usuario decidiese, pudiendo extraer la imagen original del buffer de memoria.

Se consiguió así asignar distintas resoluciones a diferentes botones dentro de la aplicación, usando tutoriales de guía [11], pero no se logró sacar el buffer con los datos.

Finalmente, y ante la circunstancia de que la resolución 227x227 era demasiado pequeña y podía no resultar cómoda para la visión del usuario, se optó por capturar la ventana de Unreal con librerías de C++ en entorno Windows y redimensionar la imagen en código a 227x227. Este redimensionado mediante código consiste en obtener 227x227 píxeles de la imagen, equitativamente distribuidos, tenga esta el tamaño que tenga de inicio.

6.3 Cambio de lenguaje

Para realizar el código de la parte del cliente, se tuvo que reescribir en el lenguaje python. El principal motivo por el cual se cambió el lenguaje fue el de facilitar la implementación del video streaming. Aunque esto implicó empezarlo de cero, el haberlo escrito previamente en lenguaje C++ aceleró mucho el proceso.

6.4 Cambios en la planificación

La planificación original que se realizó al inició del proyecto (véase apéndice 2) no se pudo seguir al completo.

Cuando la fase de finalización del prototipo acabó faltaba aproximadamente un tercio del trabajo y se tuvo que acortar la fase de expansión para acabarlo.

Este problema fue una consecuencia directa de los problemas anteriormente mencionados y a causa de esto no se implementaron peatones y pasos de cebra.

7 HERRAMIENTAS DEL PROYECTO

7.1 Unreal Studio 4.12.5

El uso de este programa fue un requerimiento por parte del tutor, dado que el Machine Learning con el que él y su grupo de investigación trabajan será eventualmente trasladado de unity3d a Unreal Engine y este proyecto podrá, en ese caso, ser usado de referencia.

Si hubiéramos elegido una herramienta con la que hubiese trabajado previamente, posiblemente el nivel prototipo habría contenido más funcionalidades y sería más completo, pero considero que aprender Unreal Studio ha sido interesante para mi formación, dada su complejidad y el tipo de productos de alto nivel profesional que han sido creados con ella.

Básicamente no era la opción óptima para el tiempo del que se disponía, pero dados los intereses que teníamos ambas partes y el hecho de que no se trate de un proyecto comercial, opino que ha sido la decisión más inteligente.

7.2 Visual Studio 2015

Visual studio es la herramienta que usa Unreal Studio por defecto si se desea trabajar en código C++ y dado que estoy familiarizado con ella, por haberla utilizado a lo largo de los estudios de ingeniería, he considerado que era la adecuada para la realización de este proyecto.

7.3 Python 2.7 Anaconda distribution

Usé la distribución 2.7 de python porque quería hacer un algoritmo pequeño y estable, así que opté por no arriesgarme con distribuciones más recientes, que pudiesen ocasionar problemas imprevistos.

He usado la versión Anaconda, que simplemente viene con las librerías que necesitaba incluidas, pero alguien sin esa versión puede ejecutar mi script siempre que también instale las librerías adecuadas.

7.4 OpenOffice

Programa utilizado para confeccionar los informes.

7.5 Paint

Programa utilizado para recortar las capturas de pantalla.

7.6 Nclass

Nclass es un programa sencillo para crear diagramas de clases, con el cual he ido actualizando mi propio diagrama a lo largo del semestre.

7.7 GanttProject

Usé este programa para crear un diagrama de Gantt para el informe inicial (véase apéndice 2).

7.8 Fraps

Para testear los frames por segundo (FPS) de la aplicación en la fase final utilicé el programa Fraps, que aunque sirve principalmente para grabar aplicaciones y crear vídeos, superpone un contador de FPS a cualquier aplicación que esté en primer plano.

8 Resultados y conclusiones

ESTE proyecto se ideó como la creación de un prototipo para trabajar machine learning aplicado a vehículos autónomos en un entorno Unreal Engine. Una primera toma de contacto, que pudiese mostrar las dificultades al implementar un proyecto así a gran escala.

Teniendo en cuenta esto, los resultados han sido satisfactorios tanto a la hora de demostrar que generar un proyecto así es viable, como a la de dejar ver las metodologías más eficientes y los problemas más frecuentes que pueden darse.

Para analizar esto dividiré las conclusiones en diferentes módulos tras analizar sus resultados.

Velocidad

La aplicación funciona a unos 30 FPS emitiendo 6 imágenes por segundo a través del socket.

La aplicación inicial no tenía ninguna latencia perceptible hasta que se implementaron los sockets y la generación de imágenes.

Los sockets se lograron optimizar hasta que la latencia fuera imperceptible. No obstante se testeo a nivel local, habría que pensar como se puede mejorar esta parte si se pretenden abarcar distancias largas entre servidor y cliente.

La generación de imágenes ha quedado como el mayor bottleneck de la aplicación, no permitiendo que esta suba de 30 FPS si no se baja la frecuencia de generación de imágenes. Mi conclusión es que, para lograr mayor velocidad, esta sería la parte que más debería ser atacada en términos de optimización.

Cabe destacar que la funcionalidad de video streaming da indicios de ser el siguiente bottleneck, pero al ser completamente dependiente de la generación de imágenes nunca llega a convertirse en el problema principal. También se le dio menos importancia ya que su funcionalidad es puramente de testing.

Si se implementase, por el motivo que fuera, cabría informarse de cómo crear un video streaming siguiendo protocolos ya establecidos que lo optimicen al máximo.

Entorno 3D

Al final se han incluido cruces, semáforos y coches que conducen bien gracias a lógica hardcoded.

Cuando empecé me habría gustado implementar peatones y pasos de cebra, pero las otras funcionalidades tenían más prioridad y con los actores con los que ya contaba el entorno 3D era suficiente para crear un prototipo, así que tuve que seguir adelante sin ellos.

El mayor obstáculo fue aprender no solo a usar Unreal Engine sino a usarlo mientras se programa el código C++.

La filosofía Unreal parece estar orientada a grupos de trabajo que incluyan diseñadores e informáticos trabajando en paralelo. Los informáticos crean la lógica de las clases, y los diseñadores rellenan los objetos con estas lógicas internas, sin tener que mirar el código.

No obstante, Unreal también dispone de un lenguaje intermedio llamado Blueprints, que permite acceder a variables y/o funciones tanto diseñadas por programadores como originales de las librerías Unreal.

La utilización de blueprints es esencial a nivel básico, para enlazar la lógica y los modelos, pero mi filosofía, una vez superada la fase de aprendizaje y aprendido a trabajar en C++ integrado en Unreal, ha sido la de implementar el mínimo código posible en blueprints.

El problema que le veo a los blueprints es el siguiente: Un programador usando únicamente blueprints siempre se sentirá limitado y tendrá que acceder al código C++ para crear funcionalidades complejas, por lo que es muy probable que acabe adoptando una filosofía parecida a la mía.

Por otro lado, para un diseñador/artista no será óptimo programar, aunque el lenguaje blueprint sea sencillo, para implementar funcionalidades muy simples que a un programador no le costarían en absoluto.

Por todo ello mi recomendación sería que los diseñadores tan solo supieran cómo hacer que un modelo herede de una clase C++, lo cual es un proceso rápido que no implica saber programar, y los programadores implementaran todas las funcionalidades en C++, incluidas las más sencillas.

Creo no obstante que los blueprints pueden ser increíblemente útiles para individuos o grupos que no dispongan de habilidades avanzadas de programación. Es seguramente el objetivo principal con el que fueron creados, ya que Unreal Engine es una herramienta que promueve que desarrolladores independientes creen sus propios proyectos e incluso da la opción al crear uno de prescindir de la posibilidad de acceder al C++ durante todo el desarrollo.

Sistema de sockets y coche machine learning

El coche es controlable a distancia y los sockets envían toda la información deseada.

La comunicación ahora mismo es bidireccional por un solo canal, lo que significa que el control y el envío de imágenes pasan por el mismo camino. Como no ha dado problemas no se han mirado otras opciones, pero es posible que al escalar el proyecto cupiera plantearse ejecutar dos sockets en paralelo, uno para cada funcionalidad.

Tampoco se ha tenido en cuenta la cantidad de imágenes que necesitará un algoritmo machine learning para trabajar, simplemente se ha intentado enviar el mayor número posible de imágenes.

Pero, por supuesto, si esto se traslada a una situación real enviar 30 imágenes por segundo o más puede resultar en muchas imágenes similares y potencialmente redundantes. Cabría tener en mente esto y testear diferentes frecuencias una vez se implemente una IA al otro lado del socket.

Agradecimientos

LA presentación de este proyecto supone el cierre de una etapa académica en la que, como siempre que hago algo que me importa, he gozado de la ayuda de muchas personas, a las cuales quiero agradecer su colaboración, sus ánimos y su apoyo incondicional.

Realizar el trabajo que queda reflejado en este informe ha supuesto un período de esfuerzo concentrado y a veces febril, que se ha convertido en un placer gracias a la presencia y el acompañamiento de todos ellos.

Quiero agradecer en primer lugar a Antonio M. López, mi tutor de proyecto, la oportunidad que me ha dado, no solo por ofrecerme un tema para el trabajo que me permitiera poner en práctica lo aprendido durante mi formación universitaria, sino también, especialmente, por su generosidad al trasmitirme tanto su conocimiento como su experiencia en nuestras reuniones semanales. Ha sido un auténtico placer trabajar bajo su tutela.

Gracias también a todos los profesores que he tenido durante estos años, por haber despertado en mí el interés por los diferentes campos de la informática. Ellos han prendido la llama que yo deberé mantener.

No quiero olvidarme de mis compañeros. El propio conocimiento se enriquece con la confrontación de ideas y puntos de vista, así como con la colaboración y el respeto a las opiniones del otro. Mi gratitud a todos los que me permitieron aprender con ellos.

A mis amigos les agradezco su comprensión en los malos momentos y, sobre todo, el haber compartido conmigo inquietudes y enfoques.

Y, por último, quisiera dar las gracias a mi familia.

A mi madre, por su ejemplo académico y su constante ayuda. Sin ella no hubiese podido llegar hasta aquí.

A mi padre, por su paciencia y calma infinitas, que me han inspirado a seguir adelante.

A mi abuela por apoyarme siempre y por seguir preguntándome "¿cómo va el cole?" cada día que me veía... hasta el día de hoy.

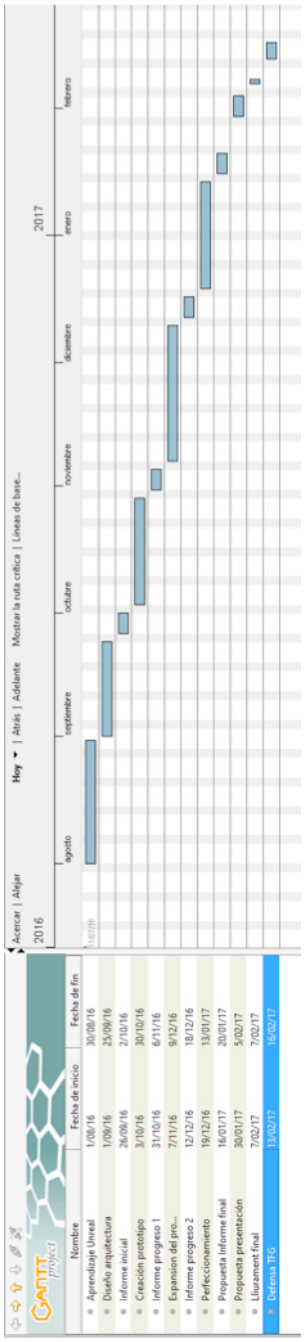
Y, especialmente, a mi abuelo, por demostrarme hasta dónde puede llegar alguien si se lo propone, sin importar el punto desde el que uno parta. Sé que se hubiera sentido orgulloso de compartir conmigo este momento.

BIBLIOGRAFÍA

[1] https://www.youtube.com/playlist?list=PLZlv_N0_O1gaCL2XjKluO7N2Pmmw9pvhE
[2] https://docs.unrealengine.com/latest/INT/index.html
[3] https://answers.unrealengine.com/index.html
[4] http://karpathy.github.io/2016/05/31/rl/
[5] https://drive.google.com/file/d/0B_wzP_JlVFcKdDg4Yy1XQTBZLUhGTG5tT29reXdYcXdES1lv/view
[6] Coursera's course on machine learning. Stanford University. 2016-2017. Enlace: https://www.coursera.org/learn/machine-learning/home/welcome. En curso.
[7] https://en.wikipedia.org/wiki/Rotation_of_axes
[8] Ejemplo de socket en Windows: https://msdn.microsoft.com/en-us/library/windows/desktop/ms737889(v=vs.85).aspx
[9] Gdi+ for testing the server: https://msdn.microsoft.com/en-us/library/windows/desktop/ms533799(v=vs.85).aspx
[10] Fscreenshot: https://docs.unrealengine.com/latest/INT/API/Runtime/Engine/FScreenshotRequest/RequestScreenshot/index.html
[11] Changing resolution of the program: https://www.youtube.com/watch?v=SPzjL-j4Aas

APÉNDICE

A1. PLANIFICACIÓN



A2. DIAGRAMA DE CLASES