INDEX

PANZ Development:
Marisa Risueño, Pablo Requena, Marcos González

MAKING OF

The beginning of this development started at the first Automatic Reasoning class, an Computer Engineering subject, career which is cursing by the three components of PanZ. We had to design and develop a videogame with CPCtelera, including artificial intelligence, and submit it at CPCRetroDev 2016.

We were thinking about making a Pokemon style videogame, or a videogame with artificial intelligence minigames, like Tic Tac Toe or stone-paper-scissors, etc. But, finally, we decided to make an effort and implement a CPC Fat Ball, which essentially is an ice soccer game.

At first, we started to work with some CPCtelera basic functions, like sprite printing, working with data structs, using tilemaps, moving our characters, etc. After some iterations, project started to look like a videogame.

At this moment, we had some hard decisions ahead, for example, how to focus our artificial intelligence, how physics will work, etc. Artificial Intelligence should be applied by fuzzy logic, because of AI need to react progressively to little changes at game status. Initial physics started like energies exchange approach, where friction force and collision were calculated with friction coefficient and characters' mass. Along those iterations, we did some design tests, like test some maps or different sprites.

At this point, AI and physics were working, and some design fitting after, our videogame was playable, but it went so lowly. That fact was due to using CPCtelera float type (f32), type which Amstrad isn't designed for and we needed because of fractional calculations. f32 using drove us to around 90% slowdown comparing it with non-applying those operations.

We could solve it using integer logic, and applying a 256 scale on those ones, which first byte represents the integer part, and second byte represents the fractional part, and we could split those with scale division. The secret here is applying two power numbers, which CPU will operate with an easy displacement operation.

Comparing both techinques, new calculation time was kind of ridiculous, then our game was really playable, but we had one week left to CPCRetroDev delivery and we had no content.

At this last iteration, we encoded a menu, different AI's difficulties, classic tournament mode, and we spent a lot of time at design and sprite/tilemap tasks. We included music too, and after thinking and working around the project, it became Pingu Soccer, the game we are submitting to CPCRetroDev 2016.

¿CÓMO SE HIZO?

El comienzo del desarrollo de este videojuego comenzó el primer día de clase de Razonamiento Automático, una asignatura de Ingeniería Informática, carrera que estudiamos los tres componentes de PanZ. Debíamos diseñar y desarrollar un videojuego con CPCtelera, con una inteligencia artificial, y presentarlo al CPCRetroDev 2016.

Estuvimos pensando en hacer un juego al estilo Pokémon o un juego de minijuegos con inteligencia artificial, como tres en raya, piedra, papel, tijera, etc. pero finalmente, decidimos apostar fuerte e implementar un Fat Ball, que, básicamente, es un juego de futbol sobre hielo.

En un principio, comenzamos a trabajar con las funciones básicas de CPCtelera, pintar sprites, crear estructuras de datos y trabajar con ellas, utilizar tilemaps, mover a los personajes, etc. Tras algunas iteraciones, el proyecto empezaba a tener un poco de pinta de videojuego.

En este momento, pasamos a tomar algunas decisiones como por ejemplo, cómo íbamos a enfocar la inteligencia artificial, las físicas, etc. La IA debía ser aplicada por lógica difusa (fuzzy), dado que el comportamiento de la IA debía reaccionar progresivamente a pequeños cambios en el estado del juego. Las físicas iniciales comenzaron con un planteamiento de intercambio de energías, en la cual, la fuerza de rozamiento y las colisiones, se calculaban en base al coeficiente de fricción y a la masa de los personajes. Durante estas iteraciones, se hicieron algunas pruebas de diseño, como probar algunos mapas y sprites diferentes.

Llegado este punto, la IA y las físicas funcionaban, y tras algunos ajustes de diseño, parecía que era jugable, pero funcionaba lentísimo. Esto se debía a que, tanto la IA como las físicas, utilizaban número fraccionarios, y para calcularlos, utilizamos el tipo float de CPCtelera (f32), para cuyos cálculos no está preparado un AmstradCPC, así que debe descomponer cada f32 en diferentes registros, operarlos individualmente, y luego juntar el resultado. Esto nos llevó a una ralentización aproximada del 90% de la velocidad del juego sin estas operaciones.

Pudimos solucionar este problema trabajando con u16 e i16 y lógica entera, es decir, aplicándoles una escala de 256 para mantener en un número entero, en el primer byte la parte entera, y en el segundo, la parte fraccionaria, que podemos obtener al dividir por la escala. El secreto es aplicar una escala potencia de 2, ya que el procesador sólo realiza un sencillo desplazamiento.

Comparando las dos técnicas, el tiempo que tardaban nuestras nuevas operaciones era ridículo, por lo que el juego era realmente jugable, pero quedaba una semana para la entrega del videojuego y no teníamos contenido.

En esta última iteración, programamos un menú, varias dificultades para la IA, y un modo torneo bastante sencillo, se invirtió mucho tiempo en tareas de diseño y aprovechamiento de sprites/tilemaps. También incluímos la música, y tras darle alguna vuelta de tuerca, llegó a ser el Pingu Soccer, el juego que estamos entregando al CPCRetroDev 2016.