Introducción a la Programación de Videojuegos

Trabajo Práctico Individual

“Minion 2”

Universidad Nacional de Quilmes

2018

Consigna

El segundo trabajo práctico individual consiste en implementar un juego sencillo que incorpore algunas de las mecánicas vistas en la cursada, utilizando o bien LibGDX, o bien la misma tecnología que se está usando para el Boss.

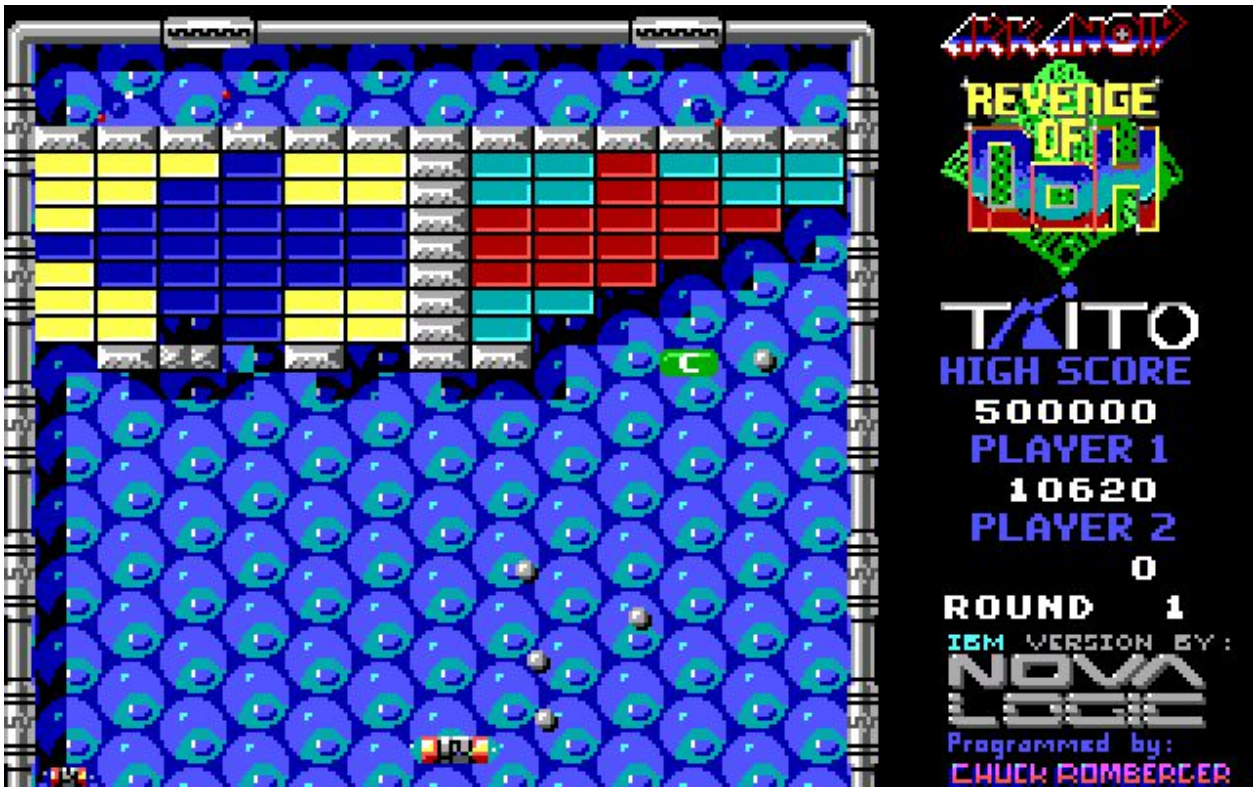
Para esto, se proponen varias alternativas. **El alumno deberá elegir una**:

- **Platformer:** Implementar un prototipo de juego de plataformas 2D estilo Super Mario o Megaman.
    - El juego debe utilizar la técnica de tiling para las plataformas, tener algunos obstáculos para sortear (precipicios, por ejemplo) y al menos un enemigo que debe poderse matar de alguna manera (disparándole, saltándole, o con alguna otra mecánica). Pueden usarse los tiles de cualquiera de los siguientes: https://opengameart.org/content/2d-platformersidescroller-tiles
    - Los datos del nivel (es decir, la especificación de qué tipo de bloque va en qué lugar, dónde está cada enemigo deben estar definidos por separado del resto del código. Esto es, o bien usando un DSL interno en el código, o bien en un archivo externo.
    - El personaje principal debe estar animado. Puede usarse el arte de cualquiera de los siguientes https://opengameart.org/content/platformersidescroller-characters-and-enemies
    - Finalmente, al matar el enemigo debe haber una probabilidad de que este dropee una moneda o un power up, que debe poder recogerse.

- **Batalla JRPG:** Implementar un prototipo de juego que consista en la pantalla de combate de un JRPG, al estilo Final Fantasy, Chrono Trigger, Phantasy Star, entre otros.
    - El juego debe contar con dos bandos: El jugador y la computadora
    - El jugador debe controlar al menos 2 personajes distintos.
    - La computadora puede ser un único enemigo o varios.
    - El juego debe desarrollarse por turnos, alternándose entre turno del jugador y la computadora. Alternativamente, el turno puede ser del personaje.
    - En el turno, el jugador debe poder elegir una de varias opciones: Atacar físicamente, con magia, con habilidad especial, usar ítem, etc. Al menos 3 opciones distintas, con efectos distintos.
    - Cada personaje debe tener sus stats particulares que afectan el resultado de su acción. Stats como fuerza mágica, fuerza física, defensa mágica, defensa física, destreza, inteligencia, etc. Idealmente, cada personaje debe tener también una habilidad única. Como ejemplo, puede consultarse la guía de stats del Ragnarok Online: http://irowiki.org/wiki/Stats. No hace falta que sea tan complejo, el sistema, pero sirve como referencia.
    - El sistema de combate debe incluir algunos elementos aleatorios: Daño aleatorio, probabilidad de falla, probabilidad de golpe crítico, de muerte súbita, etc.
    - El o los oponentes deben tener al menos dos tipos de ataque cada uno.


WHISTLE
COMD
MACR
RUN
Gryz
HP: 67
TP: 24
Chaz
HP: 44
TP: 34
Alys
HP: 35
TP: 49
Hahn
HP: 36
TP: 64
Rika
HP: 37
TP: 16
NINTENDOLIFE.COM

- **Breakout clone:** Implementar un clon de Breakout incluyendo varios niveles, features y un editor de niveles
    - El juego debe ser un clon lo más completo posible del juego breakout o arkanoid
    - Debe incluir la paleta, la pelota y varios tipos de ladrillo: Algunos rompibles, otros irrompibles. Algunos con varios golpes, otros con uno.
    - Debe incluir 5 power ups distintos. Al romper un ladrillo debe haber una chance de que un power up caiga. Los power ups a implementar son:
        - Laser: Permite disparar lasers desde la paleta, temporalmente.
        - 2 pelotas: Duplica la pelota. El jugador pierde una vida cuando todas las pelotas caen por debajo de la pantalla.
        - Retención de pelota: En lugar de rebotar la pelota, esta queda en la paleta hasta que el jugador toque una tecla.
        - Alargar paleta: La paleta se vuelve más grande
        - Vida extra: Se obtiene una vida extra
    - Se debe implementar además un editor de niveles sencillo, e incluir en el juego al menos tres niveles distintos. El juego debe avanzar de nivel y terminar.
    - Debe contabilizarse un puntaje y las vidas del jugador.



- **Estrategia en tiempo real:** Implementar un prototipo de juego de estrategia en tiempo real, con un mapa y varios tipos de unidades.
    - El mapa del juego puede implementarse con tiling. Debe incluir algún terreno pasable y algún terreno no pasable.
    - Puede hacerse con vista top down (sencillo) o con vista isométrica (más complicado).
    - Para arte, puede usarse algo como esto: https://opengameart.org/content/seven-kingdoms
    - Debe poder seleccionarse unidades (una o varias) y mandarlas a moverse o atacar
    - Debe haber al menos dos bandos que puedan atacarse entre sí
    - El mapa debe ser más grande que la cámara
    - Los datos del mapa deben estar claramente separados del código, ya sea en un archivo aparte o en un DSL
    - Debe haber al menos un recurso, una forma de minarlo y una forma de cambiarlo por unidades (No necesariamente a través de un edificio. Puede ser un botón del teclado)
    - Al menos dos tipos de unidades distintas.
    - Bonus: Las unidades deben moverse con path-finding

Corrección del trabajo práctico

Todas las opciones del trabajo práctico pueden requerir mucho trabajo si se intentan realizar como juegos completos (salvo quizás el breakout clone). Se busca, entonces, **no realizar juegos completos** sino implementar la mayor cantidad mecánicas core posibles, explorando qué sería necesario para construir un juego del género. Es importante priorizar los elementos detallados en el enunciado para cada alternativa.

La corrección será presencial y en clase, y se evaluará tanto la calidad del trabajo presentado (cantidad de mecánicas, pulido de mecánicas, calidad y extensibilidad del código, separación de responsabilidades, etc.) como las conclusiones que el alumno pueda enunciar al momento de la entrega.