

|  | **Carrera:** Técnico Universitario en Programación  **Materia:** Laboratorio de Computación II  **Tema:** TP Final |
| --- | --- |

**Integrantes de la cátedra**

Kloster, Daniel - Profesor

Wenner, Maximiliano - Ayudante

**TP Final Laboratorio II. 2022 1C**

**Integrantes del grupo**: Ignacio Aprile (Nicolas Verón)

**Tipo de proyecto: Videojuego**

**Nombre:** Arcano

**Tema:** Juego de supervivencia / acción multijugador.

**Descripción:** (escrito sintético de lo que se pretende que el sistema realice)

Arcano es un juego de combate que toma algunos conceptos mecánicos de los juegos de rol, específicamente del sistema de pelea de Dungeons and Dragons 5º Edición.

**MODO un jugador:**

El personaje Brian comienza su aventura en una isla flotante y deberá derrotar a los enemigos.

-Primero tendrá que vencer a Cross que se encuentra en la entrada a una pirámide.

-Segundo a Azazel en las cuevas de la isla.

-Por último, deberá aniquilar a la criatura más poderosa, un Orco llamado Thrall que se encuentra en las montañas.

En cada batalla Brian deberá intentar acabar con sus enemigos con la ayuda de los dados, y además, podrá tomar la decisión de atacar o curarse en combate con la ayuda de su arma.

**MODO dos jugadores:**

Cada jugador podrá elegir un personaje y enfrentarse en un combate al mejor de 3.

**Mecanicas del juego:**

Dados:

En este informe al igual que en el propio juego se denotan los dados con la letra “D” seguido de la cantidad de caras que tengan.

Además, si se debe tirar el dado más de una vez, delante de la letra se encuentra la cantidad de veces que se debe tirar (si no hay ningún número sólo es un dado).

**Ejemplo:** Un dado de 20 caras se denota como D20 y un dado de 6 caras que se lanza dos veces como 2D6

Realizar un ataque:

| **D20 + Tu bonificador de fuerza** | **VS** | **Resistencia del enemigo + boni   Armadura** |
| --- | --- | --- |

D20 = Significa que se tira un dado de 20 caras. (Rand).

D20 + Tu bonificador de fuerza vs Resistencia del enemigo + bonificador Armadura.

**D20 = 1:** Conocido como **pifia**, tu ataque automáticamente falla.

**D20 + Tu bonificador de fuerza < Resistencia del enemigo:** Tu ataque no surte efecto y no le ocasionas ningún daño a tu enemigo.

**D20 +Tu bonificador de fuerza ≥ Resistencia del enemigo:** Tu ataque si surte efecto y debes lanzar el daño que le causas a tu enemigo.

**D20 = 20:** Conocido como **crítico**, tu ataque automáticamente acierta y el daño se duplica

Ejemplo 1: El personaje guerrero, su ataque va a ser un D20+5. Si sale un 2 en el dado, su ataque final es 7. Si la resistencia de su enemigo es 10, significa que no logró hacer un buen ataque para causarle daño. La armadura del defensor es mayor al daño de ataque.

Ejemplo 2: Si sale un 20 en el dado, no importa la resistencia del enemigo, el personaje atacante realiza un ataque sorprendente que logra causar más daño de lo normal.

Calcular el daño :

Cada personaje tiene un arma diferente, esto significa que cada arma tiene un dado y un bonificador de daño diferente al resto.

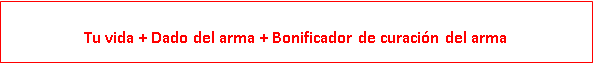


Ejemplo 1: El personaje principal tiene una espada, el dado del daño de la espada es un D8 y su bonificador de daño es un +5. Si sale un 4 en el dado, el daño que le produciría a su enemigo sería: 4+5=9.

Ejemplo 2: Si el ataque fuera un crítico serían dos dados de daño. Es decir, 2D8+5. Si saliera un 2 en el primer dado y un 5 en el segundo, el daño total sería: 2+5+5=12.

Calcular la curación:

Al igual que el daño, cada arma tiene un dado y un bonificador de curación diferente al resto.



Ejemplo: supongamos que tienes 7 de vida y en su dado del arma que es un D6 le sale un 3. Entonces, más su bonificador de curación que es un +5, la vida que le quedaría sería: 7+3+5=15.

**Archivos a utilizar:**

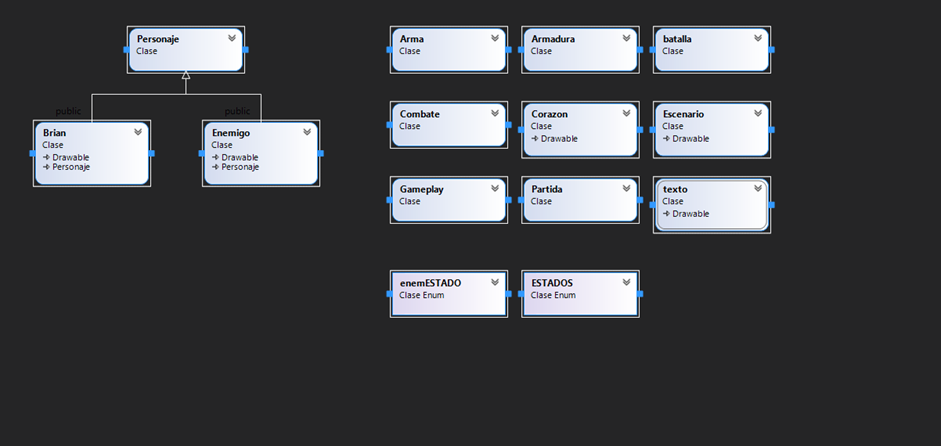
“partidasGuardadas.dat” -> guarda toda info. incluyendo en que ronda de pelea se encuentra.

“personajes.dat”-> guarda estadísticas de todos los personajes incluyendo, vida, Sprite, texturas, resistencia armas,etc.

**Clases a desarrollar:**

**Sin SFML - Aplicación consola  
**

**C/ SFML en progreso**



Personaje:

class Personaje{

protected:

///PROPIEDADES

char \_nombre[10];

int \_vida;

int \_vidaRestante;

int \_resis;

int \_boniFuerza;

Arma \_armamento;

Armadura \_armazon;

public:

///GETS

char \* get\_nombre();

int get\_vida();

int get\_vidaRestante();

int get\_resis();

int get\_boniFuerza();

Arma get\_armamento();

Armadura get\_armazon();

///SETS

void set\_nombre(char \*);

void set\_vida(int);

void set\_vidaRestante(int);

void set\_resis(int);

void set\_boniFuerza(int);

void set\_armamento(Arma);

void set\_armazon(Armadura);

///METODOS PARA ARCHIVOS

void cargar();

void mostrar();

bool escribirEnDisco();

bool leerDeDisco(int);

bool modificarEnDisco(int);

///ACCIONES EN COMBATE

int atacar(int);

int defenderse();

int realizarDanio();

void recibirDanio(int);

void curarse();

};

Brian: Heredadara de Drawable y Personajes, comportamientos y métodos.

class Brian:public sf::Drawable, public Personaje{

private:

sf::Sprite \_personaje;

sf::Texture \_texture;

ESTADOS \_estado;

float \_velocidadsalto;

float \_move;

sf::Vector2f \_posicion;

float \_frame;

public:

Brian();

void cmd();

void update();

void camino();

void combate(int);

sf::Sprite& getsprite();

void combatstance();

void animacionataque();

const sf::Sprite& checkcolision() const;

void draw(sf::RenderTarget& target, sf::RenderStates states) const override;

};

ESCENARIO:

class Escenario:public sf::Drawable{

private:

sf::Sprite \_escenario;

sf::Texture \_texture;

public:

Escenario();

//escenarios

void cambiocueva();

void cambiomontaña();

void cambiodesierto();

//batallas

void introbatalla1();

void introbatalla2();

void introbatalla3();

void batalla1();

void batalla2();

void batalla3();

//1 3 5 brian --- 2 cross --3 azazel -- 6 thrall

void decision1();

void decision2();

void decision3();

void decision4();

void decision5();

void decision6();

//menus imágenes àcambian imagen del escenario

void cambiomenu1();

void cambiomenu2();

void cambiomenu3();

void cambiomenu0();

void cambiomenupartida();

//curaciones imágenes de curaciones

void curacionb1();

void curacionb2();

void curacionb3();

void curacionb4();

void curacionb5();

void curacionb6();

//libros imágenes libros

void libro5();

void libro4();

void libro3();

void libro2();

void libro1();

//mas menus

void menupersonajes();

void menupersonajesbrian();

void menupersonajescross();

void menupersonajesthrall();

void menupersonajesazazel();

//instrucciones

void instrucciones1();

void instrucciones2();

void instrucciones3();

void instrucciones4();

//atancando

void ataquecritico1();

void ataquecritico2();

void ataquecritico3();

void ataquefallido1();

void ataquefallido2();

void ataquefallido3();

void ataqueexitoso1();

void ataqueexitoso2();

void ataqueexitoso3();

void ataqueenproceso1();

void ataqueenproceso2();

void ataqueenproceso3();

void draw(sf::RenderTarget& target, sf::RenderStates state) const override;

};

Todo controlado desde clase gameplay, que contiene las variables para iniciar la ventana gráfica, cambiar las imágenes de fondo, dibujar diferentes textos y varias funciones para tirar los dados, procesar los ataques/defensas y controlar todo.

//Parte privada Gameplay

/// PROPIEDADES

Brian \_jugador;

Enemigo \_enemigo[3];

//Encargado de llegado Brian al rectángulo iniciar el modo combate sf::RectangleShape rectángulo

//sfml

sf::RenderWindow\* window;

sf::VideoMode video;

sf::Event sfmlevent;

sf::Mutex mutex;

bool endgame;

sf::Font fuente;

sf::Text s;

int canttext;

sf::Text\* \_texto;

OPCIONES opc;

estadoEscenario background;

Estadogame est;

bool booltext;

Escenario fondo;

int cantpaglibro;

int tiempo;

int ronda;

int D;

bool boolD;

int D2;

bool enter1;

//funciones privadas de la clase

void initVariables();

void initWindow() ;

void initfuente(int pos);

void inittexto();

public:

/// CONSTRUCTOR

Gameplay();

//FUNCIONES PARA SFML

const bool running() const;

//Encargado de actualizar los inputs de teclado con eventos

void pollEvents();

//encargado de actualizar los escenarios

void escenarioUpdate();

void updateGUI();

void renderGUI(sf::RenderTarget\* target);

//Encargado de actualizar las entradas de datos

void maincmd();

//Encargado de actualizar el game loop

void update();

//Encargado de reenderizar la mayor parte del tiempo

void render();

//MUESTRA LA HISTORIA DE ARCANO

void historia();

//muestra las opciones y stats de los personajes

void MENUPERSONAJES();

bool verificacionENTER();

bool verificacionENTER2();

void vaciaropcion();

void instrucciones();

void iniciaraventura();

void classmenuupdate();

void curacionbrian( );

void curacionenem();

char IAtrucha();

void ataquebrian();

void ataqueenem();

void eleccionenem();

void fondodecisionenem();

void escenariosegunpartida();

void combatesegunpartida();

void ataquesegunpartida( );

void ataquefallidosegunpartida();

void ataqueexitososegunpartida();

void ataquecriticosegunpartida();

void resultadoataquebrian();

};