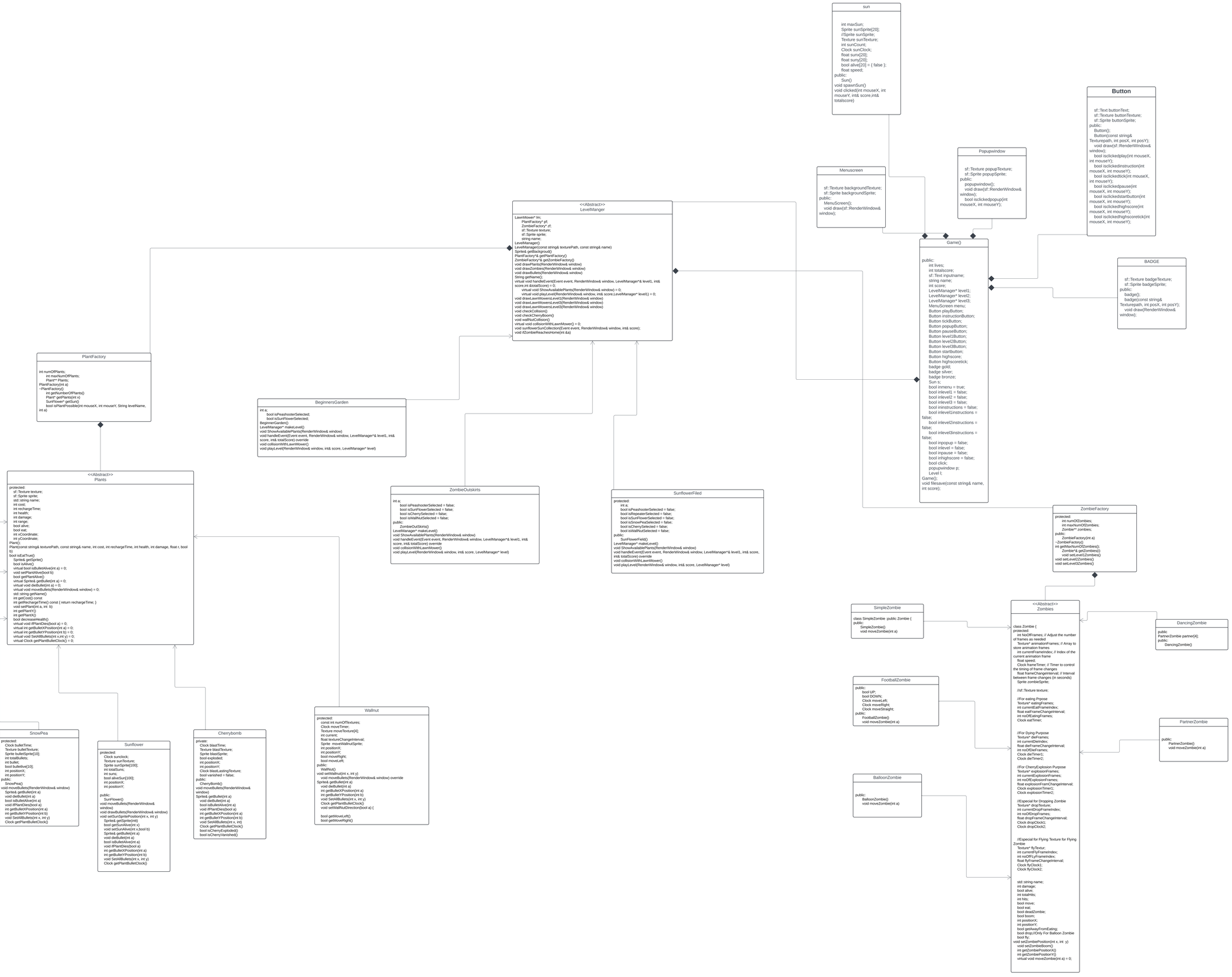
sun
int maxSun;
Sprite sunSprite[20];
//Sprite sunSprite;
Texture sunTexture;
int sunCount;
Clock sunClock;
float sunx[20];
float suny[20];
bool alive[20] = { false };
float speed;
public:
Sun()
void spawnSun()
void clicked(int mouseX, int mouseY, int& score,int& totalscore)
Button
sf::Text buttonText;
sf::Texture buttonTexture;
sf::Sprite buttonSprite;
public:
Button();
Button(const string& Texturepath, int posX, int posY);
void draw(sf::RenderWindow& window);
bool isclickedplay(int mouseX, int mouseY);
bool isclickedinstruction(int mouseX, int mouseY);
bool isclickedtick(int mouseX, int mouseY);
bool isclickedpause(int mouseX, int mouseY);
bool isclickedstartbutton(int mouseX, int mouseY);
bool isclickedhighscore(int mouseX, int mouseY);
bool isclickedhighscoretick(int mouseX, int mouseY);
Popupwindow
sf::Texture popupTexture;
sf::Sprite popupSprite;
public:
popupwindow();
void draw(sf::RenderWindow& window);
bool isclickedpopup(int mouseX, int mouseY);
Menuscreen
sf::Texture backgroundTexture;
sf::Sprite backgroundSprite;
public:
MenuScreen();
void draw(sf::RenderWindow& window);
<<Abstract>>
LevelManger
LawnMower* lm;
PlantFactory* pf;
ZombieFactory* zf;
sf::Texture texture;
sf::Sprite sprite;
string name;
LevelManager()
LevelManager(const string& texturePath, const string& name)
Sprite& getBackgroud()
PlantFactory*& getPlantFactory()
ZombieFactory*& getZombieFactory()
void drawPlants(RenderWindow& window)
void drawZombies(RenderWindow& window)
void drawBullets(RenderWindow& window)
String getName();
virtual void handleEvent(Event event, RenderWindow& window, LevelManager*& level1, int& score,int &totalScore) = 0;
virtual void ShowAvailablePlants(RenderWindow& window) = 0;
virtual void playLevel(RenderWindow& window, int& score,LevelManager* level1) = 0;
void drawLawnMowersLevel1(RenderWindow& window)
void drawLawnMowersLevel2(RenderWindow& window)
void drawLawnMowersLevel3(RenderWindow& window)
void checkCollision()
void checkCherryBoom()
void wallNutCollision()
virtual void collisionWithLawnMower() = 0;
void sunflowerSunCollection(Event event, RenderWindow& window, int& score);
void ifZombieReachesHome(int &ab)
Game()
public:
int lives;
int totalscore;
sf::Text inputname;
string name;
int score;
LevelManager* level1;
LevelManager* level2;
LevelManager* level3;
MenuScreen menu;
Button playButton;
Button instructionButton;
Button tickButton;
Button popupButton;
Button pauseButton;
Button level1Button;
Button level2Button;
Button level3Button;
Button startbutton;
Button highscore;
Button highscoretick;
badge gold;
badge silver;
badge bronze;
Sun s;
bool inmenu = true;
bool inlevel1 = false;
bool inlevel2 = false;
bool inlevel3 = false;
bool ininstructions = false;
bool inlevel1instructions = false;
bool inlevel2instructions = false;
bool inlevel3instructions = false;
bool inpopup = false;
bool inlevel = false;
bool inpause = false;
bool inhighscore = false;
bool click;
popupwindow p;
Level l;
Game();
void filesave(const string& name, int score);
BADGE
sf::Texture badgeTexture;
sf::Sprite badgeSprite;
public:
badge();
badge(const string& Texturepath, int posX, int posY);
void draw(RenderWindow& window);
PlantFactory
int numOfPlants;
int maxNumOfPlants;
Plant** Plants;
PlantFactory(int a)
~PlantFactory()
int getNumberOfPlants()
Plant* getPlants(int x)
SunFlower*.getSun()
bool isPlantPossible(int mouseX, int mouseY, String levelName, int a)
BeginnersGarden
int a;
bool isPeashooterSelected;
bool isSunFlowerSelected;
BeginnerGarden()
LevelManager* makeLevel()
void ShowAvailablePlants(RenderWindow& window)
void handleEvent(Event event, RenderWindow& window, LevelManager*& level1, int& score, int& totalScore) override
void collisionWithLawnMower()
void playLevel(RenderWindow& window, int& score, LevelManager* level)
ZombieOutskirts
int a;
bool isPeashooterSelected = false;
bool isSunFlowerSelected = false;
bool isCherrySelected = false;
bool isWallNutSelected = false;
public:
ZombieOutSkirts()
LevelManager* makeLevel()
void ShowAvailablePlants(RenderWindow& window)
void handleEvent(Event event, RenderWindow& window, LevelManager*& level1, int& score, int& totalScore) override
void collisionWithLawnMower()
void playLevel(RenderWindow& window, int& score, LevelManager* level)
SunflowerFiled
protected:
int a;
bool isPeashooterSelected = false;
bool isRepeaterSelected = false;
bool isSunFlowerSelected = false;
bool isSnowPeaSelected = false;
bool isCherrySelected = false;
bool isWallNutSelected = false;
public:
SunFlowerField()
LevelManager* makeLevel()
void ShowAvailablePlants(RenderWindow& window)
void handleEvent(Event event, RenderWindow& window, LevelManager*& level1, int& score, int& totalScore) override
void collisionWithLawnMower()
void playLevel(RenderWindow& window, int& score, LevelManager* level)
ZombieFactory
protected:
int numOfZombies;
int maxNumOfZombies;
Zombie** zombies;
public:
ZombieFactory(int a)
~ZombieFactory()
int getMaxNumOfZombies();
Zombie*& getZombies()
void setLevel1Zombies()
void setLevel2Zombies()
void setLevel3Zombies()
<<Abstract>>
Plants
protected:
sf::Texture texture;
sf::Sprite sprite;
std::string name;
int cost;
int rechargeTime;
int health;
int damage;
int range;
bool alive;
bool eat;
int xCoordinate;
int yCoordinate;
Plant();
Plant(const string& texturePath, const string& name, int cost, int rechargeTime, int health, int damage, float r, bool b)
bool isEatTrue()
Sprite& getSprite()
bool isAlive()
virtual bool isBulletAlive(int a) = 0;
void setPlantAlive(bool b)
bool getPlantAlive()
virtual Sprite& getBullet(int a) = 0;
virtual void deBullet(int a) = 0;
virtual void moveBullets(RenderWindow& window) = 0;
std::string getname()
int getCost() const
int getRechargeTime() const { return rechargeTime; }
void setPlant(int a, int b)
int getPlantY()
int getPlantX()
bool decreaseHealth()
virtual void ifPlantDies(bool a) = 0;
virtual int getBulletXPosition(int a) = 0;
virtual int getBulletYPosition(int b) = 0;
virtual void SetAllBullets(int x,int y) = 0;
virtual Clock getPlantBulletClock() = 0;
SimpleZombie
class SimpleZombie :public Zombie {
public:
SimpleZombie()
void moveZombie(int a)
<<Abstract>>
Zombies
class Zombie {
protected:
int NoOfFrames; // Adjust the number of frames as needed
Texture* animationFrames; // Array to store animation frames
int currentFrameIndex; // Index of the current animation frame
float speed;
Clock frameTimer; // Timer to control the timing of frame changes
float frameChangeInterval; // Interval between frame changes (in seconds)
Sprite zombieSprite;
//sf::Texture texture;
//For eating Prpose
Texture* eatingFrames;
int currentEatFrameIndex;
float eatFrameChangeInterval;
int noOfEatingFrames;
Clock eatTimer;
//For Dying Purpose
Texture* dieFrames;
int currentDieIndex;
float dieFrameChangeInterval;
int noOfDieFrames;
Clock deTimer1;
Clock deTimer2;
//For CherryExplosion Purpose
Texture* explosionFrames;
int currentExplosionFrames;
int noOfExplosionFrames;
float explosionFramChangeInterval;
Clock explosionTimer1;
Clock explosionTimer2;
//Especial for Dropping Zombie
Texture* dropTexture;
int currentDropFrameIndex;
int noOfDropFrames;
float dropFrameChangeInterval;
Clock dropClock1;
Clock dropClock2;
//Especial for Flying Texture for Flying Zombie
Texture* flyTextur;
int currentFlyFrameIndex;
int noOfFlyFrameIndex;
float flyFrameChangeInterval;
Clock flyClock1;
Clock flyClock2;
std::string name;
int damage;
bool alive;
int totalHits;
int hits;
bool move;
bool eat;
bool deadZombie;
bool boom;
int positionX;
int positionY;
bool getAwayFromEating;
bool drop;//Only For Balloon Zombie
bool fly;
void setZombiePosition(int x, int y)
void setZombieBoom()
int getZombiePositionX()
int getZombiePositionY()
virtual void moveZombie(int a) = 0;
DancingZombie
public
PartnerZombie partner[4];
public:
DancingZombie()
FootballZombie
public:
bool UP;
bool DOWN;
Clock moveLeft;
Clock moveRight;
Clock moveStraight;
public:
FootballZombie()
void moveZombie(int a)
PartnerZombie
public:
PartnerZombie()
void moveZombie(int a)
BalloonZombie
public:
BalloonZombie()
void moveZombie(int a)
Peashooter
protected:
Clock bulletTime;
Texture bulletTexture;
Sprite bulletSprite[100];
bool bulletLive[100];
int totalBullets;
int bullet;
int positionX;
int positionY;
Peashooter()
void moveBullets(RenderWindow& window)
Sprite& getBullet(int a)
void deBullet(int a)
bool isBulletAlive(int a)
void ifPlantDies(bool a)
int getBulletXPosition(int a)
int getBulletYPosition(int b)
void SetAllBullets(int x, int y)
Clock getPlantBulletClock()
Repeater
protected:
Clock bulletTime;
Texture bulletTexture;
Sprite bulletSprite[100];
int totalBullets;
int bullet;
bool bulletlive[100];
int positionX;
int positionY;
public:
Repeater()
void moveBullets(RenderWindow& window)
Sprite& getBullet(int a)
void deBullet(int a)
bool isBulletAlive(int a)
void ifPlantDies(bool a)
int getBulletXPosition(int a)
int getBulletYPosition(int b)
void SetAllBullets(int x, int y)
Clock getPlantBulletClock()
SnowPea
protected:
Clock bulletTime;
Texture bulletTexture;
Sprite bulletSprite[10];
int totalBullets;
int bullet;
bool bulletlive[10];
int positionX;
int positionY;
public:
SnowPea()
void moveBullets(RenderWindow& window)
Sprite& getBullet(int a)
void deBullet(int a)
bool isBulletAlive(int a)
void ifPlantDies(bool a)
int getBulletXPosition(int a)
int getBulletYPosition(int b)
void SetAllBullets(int x, int y)
Clock getPlantBulletClock()
Sunflower
protected:
Clock sunclock;
Texture sunTexture;
Sprite sunSprite[100];
int totalSuns;
int suns;
bool aliveSun[100];
int positionX;
int positionY;
public:
SunFlower()
void moveBullets(RenderWindow& window)
bool getSunAlive(int x)
void setSunAlive(int x,bool b)
Sprite& getBullet(int a)
void deBullet(int a)
bool isBulletAlive(int a)
void ifPlantDies(bool a)
int getBulletXPosition(int a)
int getBulletYPosition(int b)
void SetAllBullets(int x, int y)
void setSunSpritePosion(int x, int y)
Sprite& getSprite(int)
void setSunSpritePosion(int x, int y)
Clock getPlantBulletClock()
Cherrybomb
private:
Clock blastTime;
Texture blastTexture;
Sprite blastSprite;
bool exploded;
int positionX;
int positionY;
Clock blastLastingTexture;
bool vanished = false;
public:
CherryBomb()
void moveBullets(RenderWindow& window)
Sprite& getBullet(int a)
void deBullet(int a)
bool isBulletAlive(int a)
void ifPlantDies(bool a)
int getBulletXPosition(int a)
int getBulletYPosition(int b)
void SetAllBullets(int x, int)
Clock getPlantBulletClock()
bool isCherryExploded()
bool isCherryVanished()
Wallnut
protected:
const int numOfTextures;
Clock moveTimer;
Texture moveTexture[4];
int current;
float textureChangeInterval;
Sprite moveWallnutSprite;
int positionX;
int positionY;
bool moveRight;
bool moveLeft;
public:
WallNut()
void setWallnut(int x, int y)
void moveBullets(RenderWindow& window) override
Sprite& getBullet(int a)
void deBullet(int a)
int getBulletXPosition(int a)
int getBulletYPosition(int b)
void SetAllBullets(int x, int y)
Clock getPlantBulletClock()
void setWallNutDirection(bool a) {
bool getMoveLeft()
bool getMoveRight()