PLANTS VS. ZOMBIES™

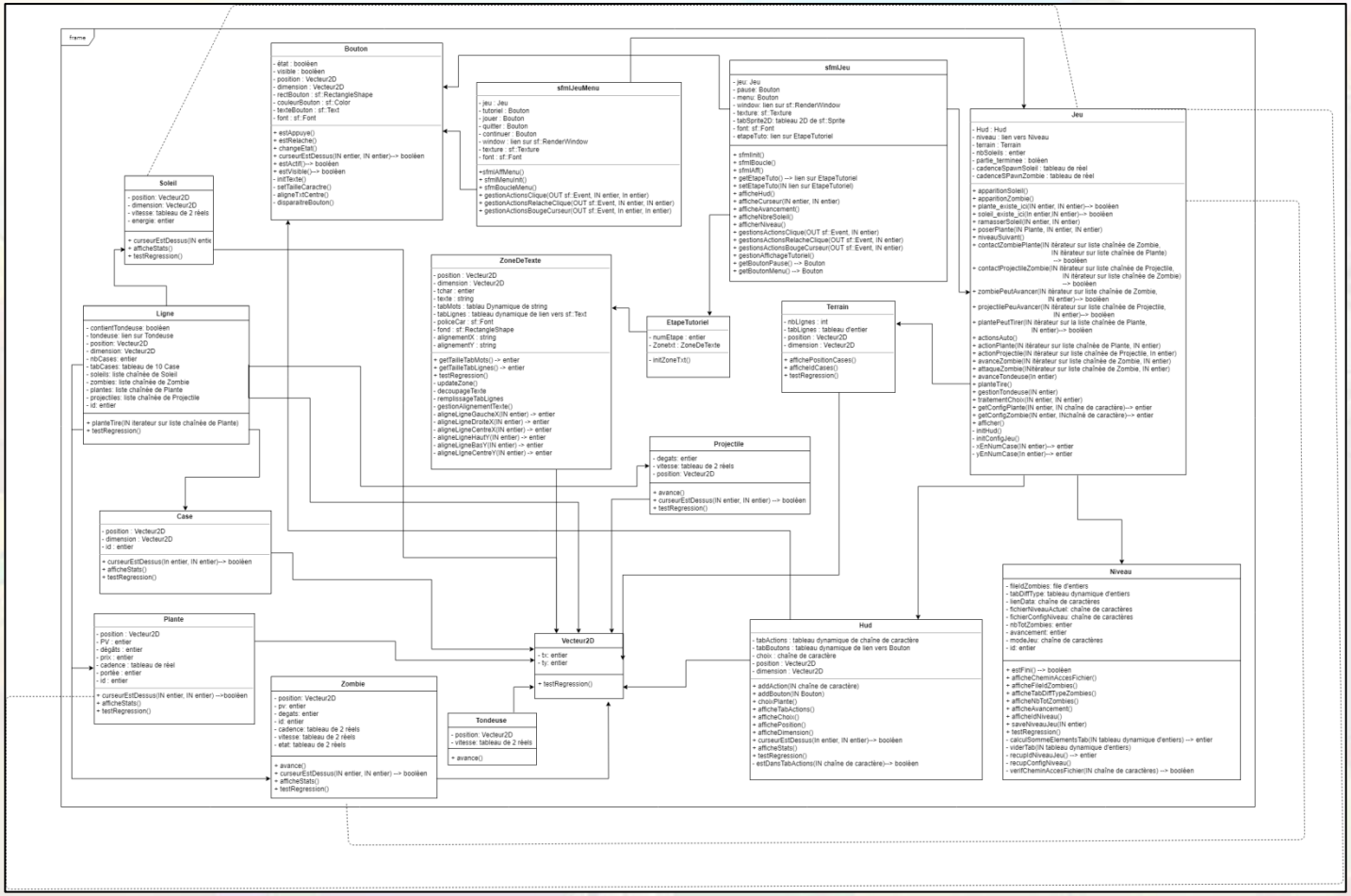
LIFAP4 - PlantVSZombies


Rémi DA COSTA


Matthieu LACOTE


Ruben GROUSSET

Diagramme des modules



17 modules

Pour

17 classes

Gestion et structure du terrain

Terrain

- nbLIgnes : int
- tabLignes : tableau d'entier
- position : Vecteur2D
- dimension : Vecteur2D

+ affichePositionCases()
+ afficheIdCases()
+ testRegression()

Ligne

- contientTondeuse: booléen
- tondeuse: lien sur Tondeuse
- position: Vecteur2D
- dimension: Vecteur2D
- nbCases: entier
- tabCases: tableau de 10 Case
- soleils: liste chaînée de Soleil
- zombies: liste chaînée de Zombie
- plantes: liste chaînée de Plante
- projectiles: liste chaînée de Projectile
- id: entier

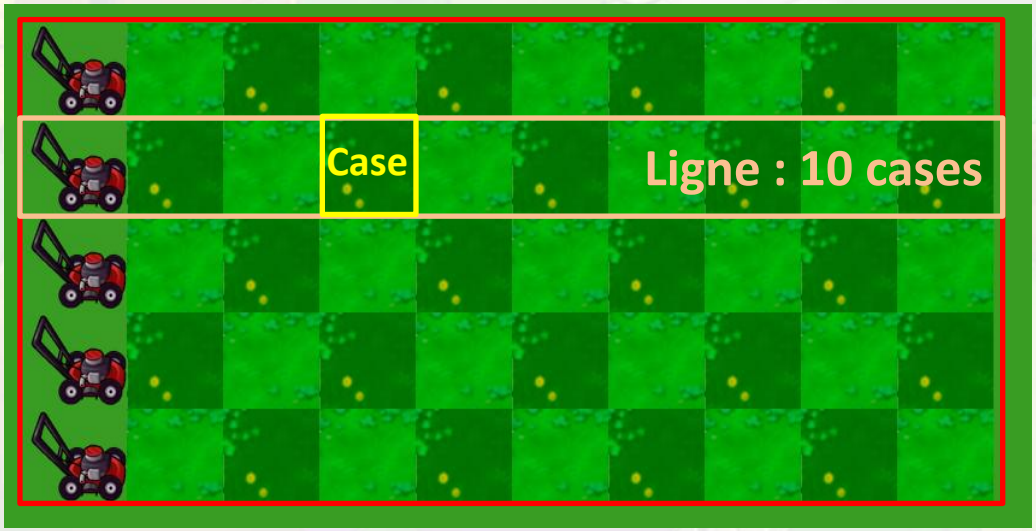
+ planteTire(IN iterateur sur liste chaînée de Plante)
+ testRegression()

Case

- position : Vecteur2D
- dimension : Vecteur2D
- id : entier

+ curseurEstDessus(In entier, IN entier)--> booléen
+ afficheStats()
+ testRegression()

Terrain : 5 lignes



Gestion et structure du terrain

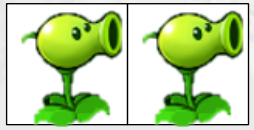
Ligne
- contientTondeuse: booléen - tondeuse: lien sur Tondeuse - position: Vecteur2D - dimension: Vecteur2D - nbCases: entier - tabCases: tableau de 10 Case - soleils: liste chaînée de Soleil - zombies: liste chaînée de Zombie - plantes: liste chaînée de Plante - projectiles: liste chaînée de Projectile - id: entier
+ planteTire(IN iterateur sur liste chaînée de Plante) + testRegression()

Listes chainées contenu dans chaque ligne :

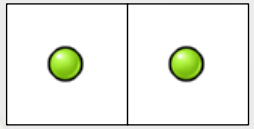
soleils

zombies

plantes



projectiles



Représentation graphique sous SFML :

Gestion du temps

Plante
- position : Vecteur2D - PV : entier - dégâts : entier - prix : entier - cadence : tableau de réel - portée : entier - id : entier

Zombie
- position: Vecteur2D - pv: entier - degats: entier - id: entier - cadence: tableau de 2 réels - vitesse: tableau de 2 réels - etat: tableau de 2 réels

Projectile
- degats: entier - vitesse: tableau de 2 réels - position: Vecteur2D

Jeu
- Hud : Hud - niveau : lien vers Niveau - terrain : Terrain - nbSoleils : entier - partie_terminee : boléen - cadenceSpawnSoleil : tableau de réel - cadenceSPawnZombie : tableau de réel

- Cadence de tir

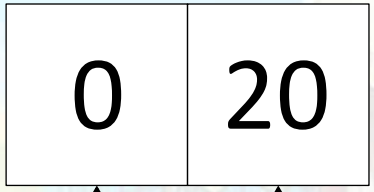
- Cadence de frappe
- Vitesse de déplacement

- Vitesse de déplacement

- Cadence Apparition Soleil
- Cadence Apparition Zombie

Gestion du temps

Structure : Tableau de 2 réels



Compteur

Borne maximale

Fonctionnement

```
Boucle de gestion du temps   // sfmlJeu //
{
   ///Incrémentation des compteurs de toutes les entitées ///
   […]
}
```

```
Boucle de gestion des actions du jeu   // Jeu //
{
   /// si les compteurs sont égaux à la borne max on redémarre le compteur et on exécute l'action ///
   […]
}
```

Module Jeu

Jeu
+ apparitionSoleil() + apparitionZombie() + plante_existe_ici(IN entier, IN entier)--> booléen + soleil_existe_ici(In entier,IN entier)--> booléen + ramasserSoleil(IN entier, IN entier) + poserPlante(IN Plante, IN entier, IN entier) + niveauSuivant() + contactZombiePlante(IN itérateur sur liste chaînée de Zombie, IN itérateur sur liste chaînée de Plante) --> booléen + contactProjectileZombie(IN itérateur sur liste chaînée de Projectile, IN itérateur sur liste chaînée de Zombie) --> booléen + zombiePeutAvancer(IN itérateur sur liste chaînée de Zombie, IN entier)--> booléen + projectilePeuAvancer(IN itérateur sur liste chaînée de Projectile, IN entier)--> booléen + plantePeutTirer(IN itérateur sur la liste chaînée de Plante, IN entier)--> booléen + actionsAuto() + actionPlante(IN itérateur sur liste chaînée de Plante, IN entier) + actionProjectile(IN itérateur sur liste chaînée de Projectile, In entier) + avanceZombie(IN itérateur sur liste chaînée de Zombie, IN entier) + attaqueZombie(INitérateur sur liste chaînée de Zombie, IN entier) + avanceTondeuse(In entier) + planteTire() + gestionTondeuse(IN entier) + traitementChoix(IN entier, IN entier) + getConfigPlante(IN entier, IN chaîne de caractère)--> entier + getConfigZombie(IN entier, INchaîné de caractère)--> entier + afficher() - initHud() - initConfigJeu() - xEnNumCase(IN entier)--> entier - yEnNumCase(In entier)--> entier

Fonctions importantes :

+ actionsAuto

  fait appel aux fonctions suivantes :

  + actionPlante
  + actionProjectile
  + actionZombie
  + avanceTondeuse
  + planteTire

+ traitementChoix

+ getConfigPlante

+ getConfigZombie

Module Jeu

Procédure actionAuto (dans la classe Jeu) :

```
void Jeu::actionsAuto()
{
    for(int i=0;i<getTerrain().getNbLignes();i++) (1)
    {
        for(list<Plante>::iterator it=getTerrain().getLigne(i).beginPlante();it!=getTerrain().getLigne(i).endPlante();it++) (2)
        {
            if((*it).getComptCadence()==(*it).getCadence()) (3)
            {
                actionPlante(it,i);
                (*it).setComptCadence(0);
            }
            else
            {
                (*it).setComptCadence((*it).getComptCadence()+1); (4)
            }
        }
```

Fonctionnement

(1) Parcours toutes les lignes

(2) Parcours toutes les listes d'entités de la ligne, ici parcours la liste de plantes

(3) Vérifie si une action doit être exécutée (**gestion du temps**)

(4) Incrémente les compteurs

Module sfmlJeu

sfmlJeu
- jeu: Jeu - pause: Bouton - menu: Bouton - window: lien sur sf::RenderWindow - texture: sf::Texture - tabSprite2D: tableau 2D de sf::Sprite - font: sf::Font - etapeTuto: lien sur EtapeTutoriel
+ sfmlInit() + sfmlBoucle() + sfmlAff() + getEtapeTuto() --> lien sur EtapeTutoriel + setEtapeTuto(IN lien sur EtapeTutoriel) + afficheHud() + afficheCurseur(IN entier, IN entier) + afficheAvancement() + afficheNbreSoleil() + afficherNiveau() + gestionsActionsClique(OUT sf::Event, IN entier) + gestionsActionsRelacheClique(OUT sf::Event, IN entier) + gestionsActionsBougeCurseur(OUT sf::Event, IN entier) + gestionAffichageTutoriel() + getBoutonPause() --> Bouton + getBoutonMenu() --> Bouton

Fonctions importantes :

+ sfmlBoucle

+ sfmlAff

+ gestionsActionsClique

+ gestionActionsRelacheClique

+ gestionActionsBougeCurseur

Module sfmlJeu

Procédure sfmlBoucle (dans la classe sfmlJeu) :

```
void sfmlJeu::sfmlBoucle () {
    Clock clockGeneral;

    while (window->isOpen())
    {
        while(!getJeu().getPartieTerminee())
        {

            Gestion du temps
            Gestion retour au Menu
            Gestion évènements
            Affichage


        }

        window->close();
        cout<<"PERDU"<<endl;

    }
}
```

```
if(!enPause())
{
    if(clockGeneral.getElapsedTime().asSeconds()>=0.02)
    {
        getJeu().actionsAuto();
        clockGeneral.restart();
    }
}
else
{
    clockGeneral.restart();
}
```

```
sfmlAff();
afficheCurseur(x,y);
gestionAffichageTutoriel();
window->display();
```

Module sfmlJeu

Procédure sfmlAff (dans la classe sfmlJeu) :

```
for(int i=0;i<jeu.getTerrain().getNbLignes();i++)                    (1)
{                                                                                    (2)
    for(list<Plante>::iterator it=getJeu().getTerrain().getLigne(i).beginPlante();it!=jeu.getTerrain().getLigne(i).endPlante();it++)
    {
        int id=(*it).getId();
        Sprite tmp=tabSprite2D[1][id-1];                             (3)

        tmp.setPosition(Vector2f((*it).getPosition().getVectX(),(*it).getPosition().getVectY()));

        window->draw(tmp);                                           (4)
    }
```

Fonctionnement

(1) Parcours toutes les lignes

(2) Parcours toutes les listes d'entités de la ligne

(3) Crée un Sprite adapté

(4) Affiche le sprite à l'écran

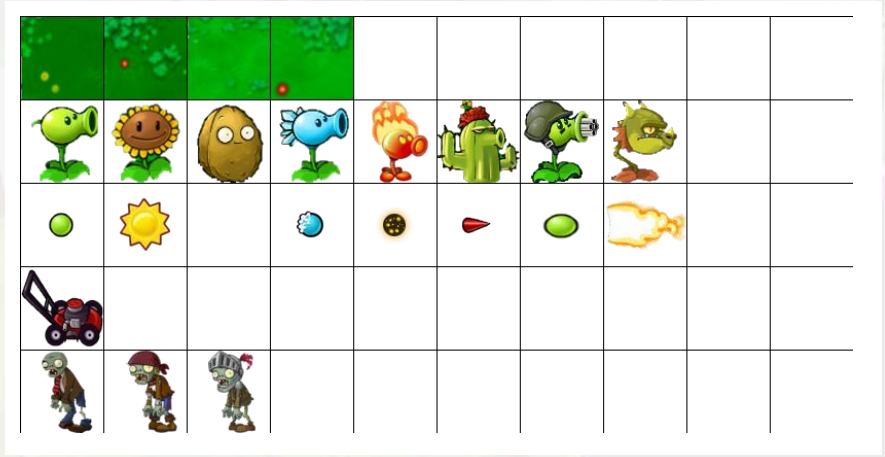


Tableau de sprites disponible dans le dossier data

Merci de votre écoute et de votre attention

Remerciements particuliers :

Elodie DESSEREE : Rapporteur de notre projet, maître de conférence à l'université Claude Bernard Lyon 1

Email : elodie.desseree@univ-lyon1.fr

Prêt pour la démonstration ?