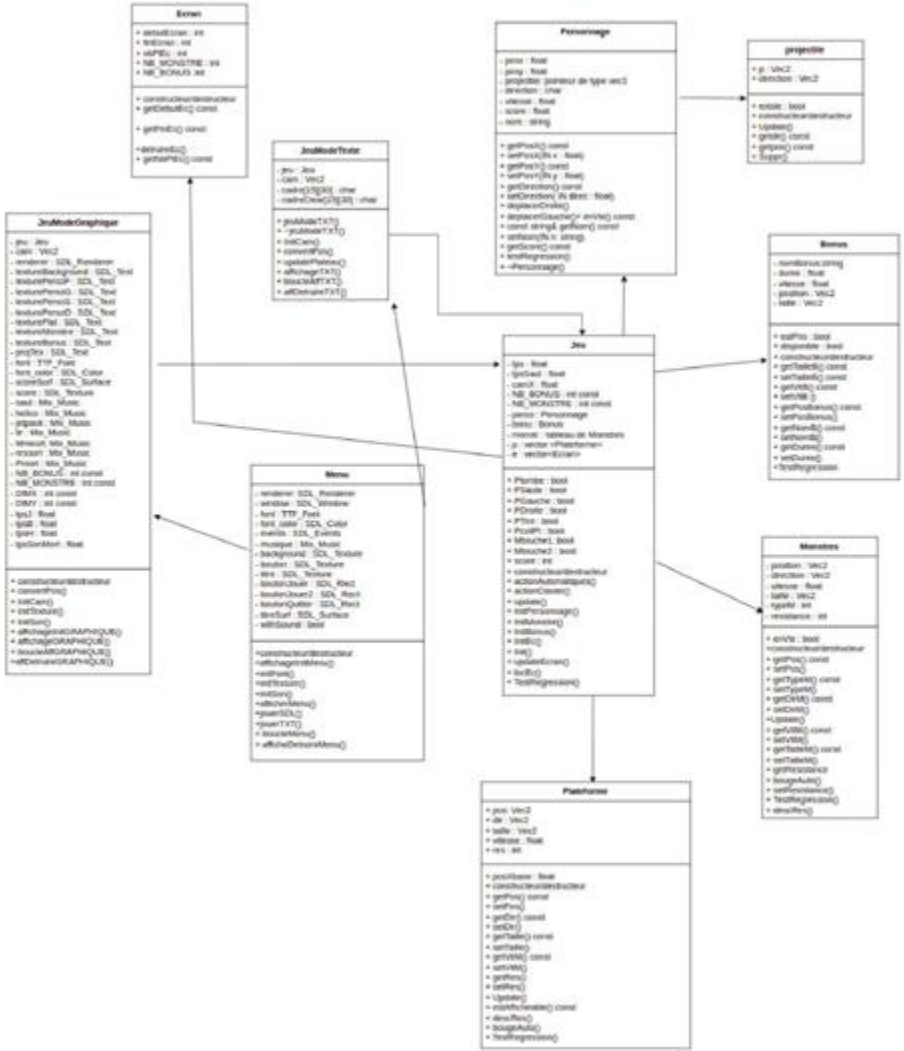
FOODLE JUMP

RIGAUT Cyril
MUNOZ Matéo


Ecran
+ ancienEcran : int
+ finEcran : int
+ vitPlEc : int
+ NB_MONSTRE : int
+ NB_BONUS :int
+ constructeur/destructeur
+ getDebutEc() const
+ getFinEc() const
+destruireEc()
+ getVitPlEc() const
Personnage
- perso : float
- ping : float
- projectile : pointeur de type vec3
- direction : char
- vitesse : float
- score : float
- nom : string
+ getPosX() const
+ setPosX(float x : float)
+ getPosY() const
+ setPosY(float y : float)
+ getDirection() const
+ setDirection(float direc : float)
+ deplacerDroite()
+ deplacerGauche(/* erreur) const
+ const string& getNom() const
+ setNom(string N n : string)
+ getScore() const
+ testRegression()
+ ~Personnage()
projectile
+ p : Vec2
+ direction : Vec2
+ existe : bool
+ constructeur/destructeur
+ Update()
+ getdir() const
+ getpos() const
+ Tirer()
JeuModeTexte
- jeu : Jeu
- clavier : Vec2
- cadre1[15][30] : char
- cadre2[15][30] : char
+ jeuModeTXT()
+ ~jeuModeTXT()
+ initCadre()
+ convertPos()
+ updatePlateau()
+ affichageTXT()
+ boucleJeuTXT()
+ effDerniereTXT()
JeuModeGraphique
- jeu : Jeu
- clavier : Vec2
- renderer : SDL_Renderer
- textureBackground : SDL_Text
- texturePersoP : SDL_Text
- texturePersoG : SDL_Text
- texturePersoS : SDL_Text
- texturePersoD : SDL_Text
- texturePlat : SDL_Text
- textureMonstre : SDL_Text
- textureBonus : SDL_Text
- projTex : SDL_Text
- font : TTF_Font
- font_color : SDL_Color
- scoreSurf : SDL_Surface
- score : SDL_Texture
- sauv : Mix_Music
- helico : Mix_Music
- jetpack : Mix_Music
- fin : Mix_Music
- tirMusic : Mix_Music
- ressort : Mix_Music
- Pivert : Mix_Music
- NB_BONUS : int const
- NB_MONSTRE : int const
- DIMX : int const
- DIMY : int const
- fps1 : float
- fps0 : float
- fpsH : float
- fpsSonMort : float
+ constructeur/destructeur
+ convertPos()
+ initCam()
+ initTexture()
+ initSon()
+ affichageGRAPHIQUE()
+ effChargGRAPHIQUE()
+ boucleJeuGRAPHIQUE()
+effDerniereGRAPHIQUE()
Jeu
- fps : float
- fpsDuel : float
- camX : float
- NB_BONUS : int const
- NB_MONSTRE : int const
- perso : Personnage
- bonu : Bonus
- monst : tableau de Monstres
- p : vector <Plateforme>
- e : vector<Ecran>
+ Pbombe : bool
+ PSaute : bool
+ PGauche : bool
+ PDroite : bool
+ PTire : bool
+ PcolPl : bool
+ MtoucheC : bool
+ MtoucheI : bool
+ score : int
+ constructeur/destructeur
+ actionAutomatique()
+ actionClavier()
+ update()
+ initPersonnage()
+ initAvancing()
+ initBonus()
+ initEc()
+ init()
+ updateEcran()
+ tirBc()
+ TestRegression()
Bonus
- nomBonus:string
- duree : float
- vitesse : float
- position : Vec2
- taille : Vec2
+ initPos :bool
+ disponible : bool
+ constructeur/destructeur
+ getTailleBc() const
+ setTailleBc() const
+ getVitBc() const
+ setVitMB ()
+ getPositionBc() const
+ setPositionBc()
+ getNomBc() const
+ setNomBc()
+ getDureeBc() const
+ setDureeBc()
+ TestRegression
Menu
- renderer: SDL_Renderer
- window : SDL_Window
- font : TTF_Font
- font_color : SDL_Color
- events : SDL_Events
- musique : Mix_Music
- background : SDL_Texture
- bouton : SDL_Texture
- titre : SDL_Texture
- boutonJouer : SDL_Rect
- boutonJouer2 : SDL_Rect
- boutonQuitter : SDL_Rect
- titreSurf : SDL_Surface
- effSound : bool
+constructeur/destructeur
+affichageImMenu()
+initFont()
+initTexture()
+initSon()
+affichImMenu()
+jouerSDL()
+jouerTXT()
+ boucleMenu()
+ effaceDerniereMenu()
Monstres
- position : Vec2
- direction : Vec2
- vitesse : float
- taille : Vec2
- typeM : int
- resistance : int
+ estVu : bool
+constructeur/destructeur
+ getPos() const
+ setPos()
+ getTypeM() const
+ setTypeM()
+ getDirMB() const
+ setDirMB()
+Update()
+ getVitMB() const
+ setVitMB()
+ getTailleMB() const
+ setTailleMB()
+ getPersistance
+ bougeAuto()
+ setResistance()
+ TestRegression()
+ detruire()
Plateforme
+ pos: Vec2
+ dir : Vec2
+ taille : Vec2
+ vitesse : float
+ res : int
+ posX/pose : float
+ constructeur/destructeur
+ getPos() const
+ setPos()
+ getDir() const
+ setDir()
+ getTaille() const
+ setTaille()
+ getVitM() const
+ setVitM()
+ getRes()
+ setRes()
+ Update()
+ estAffichelable() const
+ detruire()
+ bougeAuto()
+ TestRegression()

Classe Jeu

Classe Ecran

Classe Jeu

```
        Vec2 monstreSup;
        monstre.x = mx;
        monstre.y = my;
        monstreSup.x = mxSup;
        monstreSup.y = mySup;
        bool collisionM = doOverlap(monstreSup, monstre, posSupperso, posperso) && (monstr[i].enVie == true);
        if (collisionM && !perso.aPrisB) // si le personnage n'a pas de bonus, et si le monstre est en vie..
        {                                // on rentre en collision.
            if (Ptombe && perso.enVie)   // si le personnage est entrain de tomber
            {
                tpsSaut = 35; // on saute..
                Psaute = true;
                Ptombe = false;
                monstr[i].enVie = false; // et le monstre meurt..
                Mtouche2 = true;         // on joue le son quand on touche un monstre.
                score += 1000;
            }
            else if (!Ptombe) // sinon, on meurt.
            {
                perso.tombe(dt);
                perso.enVie = false;
            }
        }
        if (monstr[i].getResistance() == 0) // si le monstre n'a plus de resistance..
        {
            monstr[i].enVie = false; // il meurt.
        }
    }
    for (int i = 0; i < NB_BONUS; i++) // pour tous les bonus.
    {
        float bx = bonu[i].getPosBonus().x;
        float by = bonu[i].getPosBonus().y;
        float bSupx = bx - bonu[i].getTailleB().x;
        float bSupy = by + bonu[i].getTailleB().y;
        Vec2 bonus;
        Vec2 bonusSup;
        bonus.x = bx;
        bonus.y = by;
        bonusSup.x = bSupx;
        bonusSup.y = bSupy;
        bool collisionB = doOverlap(bonusSup, bonus, posSupperso, posperso) && !bonu[i].estPris && !perso.aPrisB;
        if (collisionB && perso.enVie) // si le personnage n'a pas déjà un bonus et que le bonus n'a pas déjà été pris et qu'il est vivant..
        {                              // on peut rentrer en collision.
            tps = bonu[i].getDuree();  // on initialise la duree du bonus.
            bonu[i].estPris = true;    // on indique qu'il n'est plus disponible pour le personnage.
            bonu[i].disponible = true; // on indique qu'il est disponible pour la génération de l'écran suivant.
            perso.aPrisB = true;       // on indique que le personnage a pris un bonus.
```

Classe Ecran

```
Ecran::Ecran(int posDebut, int posFin, int nbPlat, vector<Plateforme> &p, bonus b[4], Monstre m[4], bool genereMB, int scoreJeu)
{
    debutEcran = posDebut;
    finEcran = posFin;
    int difficult = scoreJeu / 1000;
    nbPlEc = nbPlat - difficult; // on ajuste le nombre de plateforme en fonction du score.
    if (nbPlEc < 15)             // si il passe en dessous de 15, on le maintient.
        nbPlEc = 15;
    Plateforme p0;
    p0.setPos(posDebut, (rand() % 11) + 1);
    for (int i = 0; i < nbPlEc; i++)
    {
        int r = rand();
        int r2 = rand();
        Plateforme tmp;                                    // on créé un tampon..
        tmp.setPos(random(posFin, posDebut), (r % 11) + 1); // on randomise la position de la nouvelle plateforme.
        tmp.posXbase = tmp.getPos().x;                     // on garde sa position verticale à la génération (pour l'intervalle de déplacement auto en vertica
        if (r % 100 < 70 - difficult)                      // 70% de chances que la resistance de la plateforme soit infinie. Ajustée en fonction du score (co
            tmp.setRes(-1);
        else // 30 qu'elle disparaisse apres un saut.
            tmp.setRes(1);
        if (r2 % 100 > 60 - difficult) // 40 % de chance qu'elle bouge horizontalement.
            tmp.setDir(0, 1);
        if (r2 % 100 > 90 - difficult) // 10 % qu'elle bouge verticalement.
            tmp.setDir(4, 0);
        tmp.setTaille(0.7, 2);
        if (genereMB) // si la génération de bonus/monstre est activée..
        {
            if (r % 100 > 80 - difficult) // pour 20% de chances..
            {
                int bonus = r % NB_BONUS; // sur le bonus tiré..
                if (b[bonus].disponible)  // si celui-ci est disponible..
                {
                    if (bonus == 4) // si c'est le trou noir
                    {
                        b[bonus].setPosBonus(random(posFin, posDebut), (r % 11) + 1); // position random dans l'écran..
                        b[bonus].estPris = false;                                     // on indique qu'il n'est pas pris.
                        b[bonus].disponible = false;                                  // on indique qu'il n'est plus disponible.
                    }
                    else
                    {
                        b[bonus].setPosBonus(tmp.getPos().x, tmp.getPos().y); // sinon on place le bonus sur la plateforme.
                        b[bonus].estPris = false;
                        b[bonus].disponible = false;
                    }
                }
            }
```

CONCLUSION

Regrets : Système d’enregistrement de score, accomplissement des tâches..

Organisation et travail d'équipe aurait pu être meilleur.