Rapport du Projet de Programmation C++

Université de Shanghai-UTSEUS-LO02

04 novembre 2018

Xu Yuhang(16124676), Chen Yiyi (16124679), Chang Xinyi (16124678)

Résumé

C'est un jeu en mode console. Le joueur déplace la barre vers la droite ou verts la gauche avec les touches correspondantes du clavier. Si toutes les briques sont cassées, le jeu est gagné.

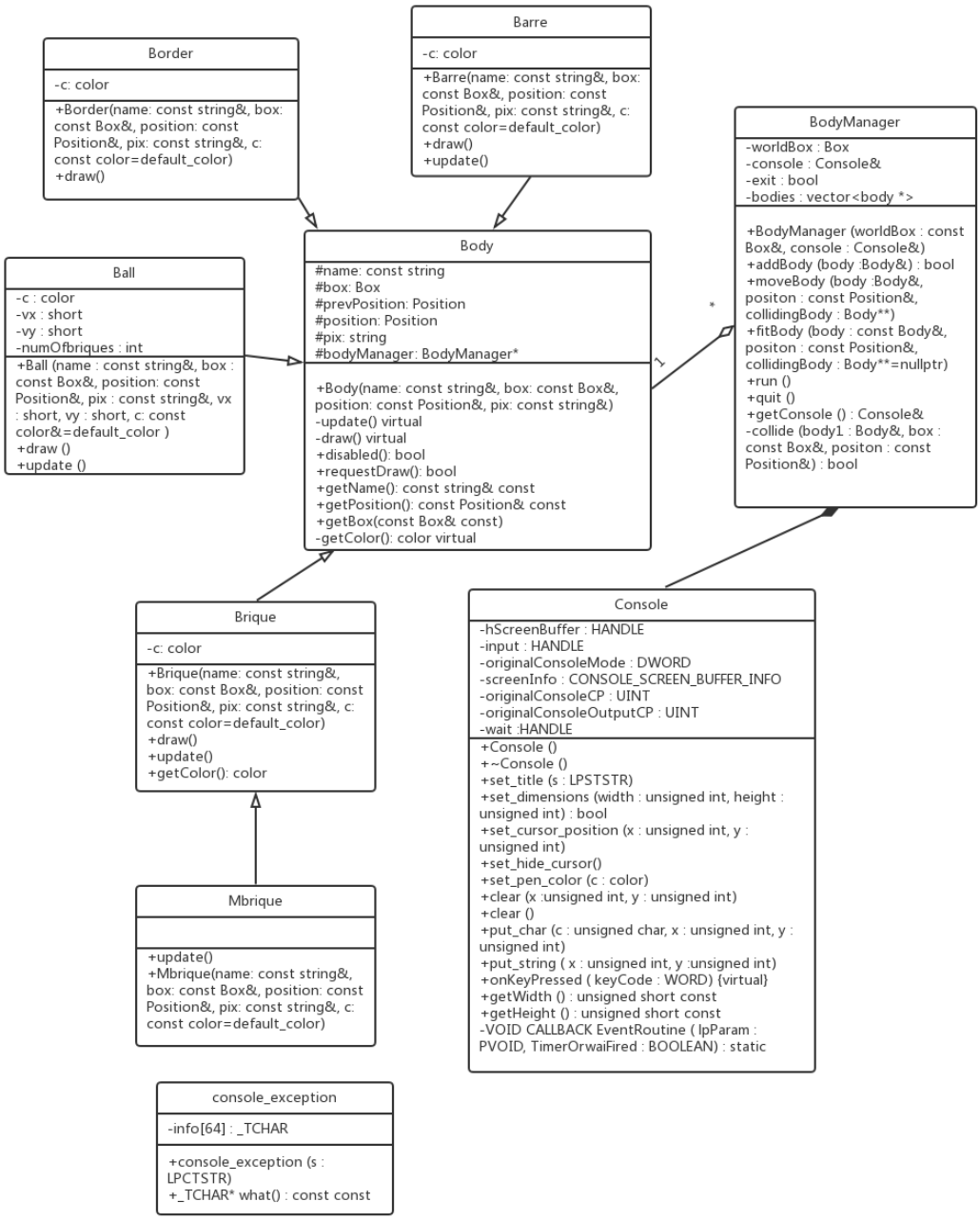
Il y a quatre types de briques :

couleur	type	nombre
Brique blanche	Brique simple	4
Brique jaune	Brique qui accélère la balle	1
Brique bleue	Brique qui est cassé après deux coups	1
Brique rouge	Brique qui se déplace	1

Les classes et UML

Il y a 4 sous-classes de la classe `Body` : **`Ball`**, **`Barre`**, **`Border`** et **`Brique`**.
Dans les classes **`Barre`**, **`Border`**, **`Brique`**, nous ajoutons l'attribut privé **`"color"`** et nous avons redéfini les méthode **`draw()`** et **`update()`** pour afficher les bodies avec couleurs différentes. Donc dans la classe de base Body, nous mettons ces méthodes virtuelles.
Dans les classes Body et Brique, nous ajoutons aussi la méthode **`getcolor()`**
La classe **`Mbrique`** est dérivée par la classe Brique. C'est pour les briques qui se déplacent. nous redéfinissons aussi la méthode **`update()`**.
Dans la classe Ball, nous ajoutons 3 autres attributs privés. **`"vx"`**, **`"vy"`** sont deux shorts qui contrôlent la vitesse et la direction de la balle. **`"numOfbriques`** "est un int qui calcule le nombre des briques cassé pour juger si le jeu est gagné.
Dans la class BodyMananger, nous modifions la méthode **`run()`** pour la briques qui accélère la balle.

C'est l'UML :



Les variables globales

Nous ajoutons 4 variables globales dans “pch.h” :

```
18  extern int judge ;
19  extern int acceler;
20  extern int numdoublebrique;
21  extern short vx1;
```

Ses fonctions va être expliquée plus tard.

Le rebond de la balle

(dans la méthode MoveBody)

```
(*pt2)->disabled = true;//brique simple //il va etre efface dans la methode update
if ((((*pt2)->getPosition().x == px) //judger si l'obstacle de rebondir est horizontale, vertical ou un coin
    && ((*pt2)->getPosition().y == py))
    || (((*pt2)->getPosition().x + ((*pt2)->getBox().width) - 1 == px)
        && ((*pt2)->getPosition().y == py))
    || (((*pt2)->getPosition().x == px)
        && (((*pt2)->getPosition().y + ((*pt2)->getBox().height)) - 1 == py))
    || (((*pt2)->getPosition().x + ((*pt2)->getBox().width) - 1 == px)
        && (((*pt2)->getPosition().y + ((*pt2)->getBox().height)) - 1 == py)))
{
    (this->vx) = -(this->vx);// repartir dans la direction opposee
    (this->vy) = -(this->vy);
}
else {
    if (((*pt2)->getPosition().x < px) && (((*pt2)->getPosition().x + (*pt2)->getBox().width - 1) > px))
        (this->vy) = -(this->vy);//l'obstacle vertical
    else
    {
        if (((*pt2)->getPosition().y < py) && (((*pt2)->getPosition().y + (*pt2)->getBox().height - 1) > py))
            (this->vx) = -(this->vx);// l'obstacle horizontale
    }
}
```

Brique simple

Si la balle rebondit sur les briques simples, le programme va juger si l'obstacle de rebondir est horizontale, vertical ou un coin. Pour les rebonds sur un obstacle horizontal ou vertical, la balle garde le même angle d'incidence. Et pour les rebonds sur des coins, la balle repart dans la direction opposée.

```
if (!(bodyManager->moveBody(*this, Position(px, py), pt2)))
{
    if ((*pt2)->getName() != "abyss")
    {
        if ((*pt2)->getName() == "brique") {//rebondir avec la brique,le compteur ajoute 1
            (this->numOfbriques)++;
            if ((*pt2)->getColor() == yellow)//Brique qui accelere la balle
            {
                (*pt2)->disabled = true;
                if ((((*pt2)->getPosition().x == px)
                    && ((*pt2)->getPosition().y == py))
                    || (((*pt2)->getPosition().x + ((*pt2)->getBox().width) - 1 == px)
                        && ((*pt2)->getPosition().y == py))
                    || (((*pt2)->getPosition().x == px)
                        && (((*pt2)->getPosition().y + ((*pt2)->getBox().height)) - 1 == py))
                    || (((*pt2)->getPosition().x + ((*pt2)->getBox().width) - 1 == px)
                        && (((*pt2)->getPosition().y + ((*pt2)->getBox().height)) - 1 == py)))
                {
                    (this->vx) = -(this->vx);
                    (this->vy) = -(this->vy);
                }
                else {
                    if (((*pt2)->getPosition().x < px) && (((*pt2)->getPosition().x + (*pt2)->getBox().width - 1) > px))
                    {
                        (this->vy) = -(this->vy);
                    }
                    else
                    {
                        if (((*pt2)->getPosition().y < py) && (((*pt2)->getPosition().y + (*pt2)->getBox().height - 1) > py))
                            (this->vx) = -(this->vx);
                    }
                }
                acceler=1;//le parametre de la methode de «sleep» se diminue, la vitesse grandit
```

Brique qui accélère la balle

Pour les rebonds sur la brique qui accélère la balle, la balle repart plus vitement. Ici on utilise une variable globale **«acceler»**. Si la balle rebondit la brique jaune, la variable **«acceler»** va égaler 1, donc le paramètre de la méthode de **«sleep»** va se diminuer et la vitesse va grandir.

```
if ((*pt2)->getColor() == blue) {//Brique qui est cassee apres deux coups
    if(numdoublebrique ==1) (*pt2)->disabled = true;//juger si c'est la deuxieme foi
    else numdoublebrique++;
    if ((((*pt2)->getPosition().x == px) && ((*pt2)->getPosition().y == py))
        || (((*pt2)->getPosition().x + ((*pt2)->getBox().width) - 1 == px)
            && ((*pt2)->getPosition().y == py))
        || (((*pt2)->getPosition().x == px)
            && (((*pt2)->getPosition().y + ((*pt2)->getBox().height)) - 1 == py))
        || (((*pt2)->getPosition().x + ((*pt2)->getBox().width) - 1 == px)
            && (((*pt2)->getPosition().y + ((*pt2)->getBox().height)) - 1 == py)))
    {
        (this->vx) = -(this->vx);
        (this->vy) = -(this->vy);
    }
    else {
        if (((*pt2)->getPosition().x < px) && (((*pt2)->getPosition().x + (*pt2)->getBox().width - 1) > px))
        {
            (this->vy) = -(this->vy);
        }
        else
        {
            if (((*pt2)->getPosition().y < py) && (((*pt2)->getPosition().y + (*pt2)->getBox().height - 1) > py))
                (this->vx) = -(this->vx);
        }
    }
```

Brique qui est cassée après deux coups

Pour les rebonds sur la brique bleue, la brique va casser après deux coups. Ici on utilise une variable globale **«numdoublebrique»** qui égale 0 par défaut. Si la brique reçoit un coup, **«numdoublebrique »** va se plus 1. S'il égale 2, la brique va être cassée.

```
else {//Brique qui se deplace
    Body* pt1 = this;
    Body** pt2 = &(pt1);
    unsigned short px = position.x;
    unsigned short py = position.y;
    unsigned int n = 0;
    px = px + vx1;//mouvement dans la direction horizontale
    if (!(bodyManager->moveBody(*this, Position(px, py), pt2)))
        if (((*pt2)->getName() == "leftBorder") || ((*pt2)->getName() == "rightBorder"))
            vx1 = -vx1; //repartie dans la direction opposee.
        else if ((*pt2)->getName() == "ball")
            disabled = true;//rebondir avec le balle
}
```

Brique qui se déplace

Pour les rebonds de la balle sur la brique qui se déplace, nous créons une variable globale **«vx»** qui représente la position prochaine dans la direction horizontale. Pour les coups sur les frontières, il va repartir dans la direction opposée.

```
else {//rebondir le border
    if ((*pt2)->getName() != "barre") {
        if (((*pt2)->getName() == "leftCoin") || ((*pt2)->getName() == "rightCoin")) {
            if (((*pt2)->getPosition().x == px) && ((*pt2)->getPosition().y == py))
            {
                (this->vx) = -(this->vx);
                (this->vy) = -(this->vy);
            }
        }
        else
            if (((*pt2)->getName() == "leftBorder") || ((*pt2)->getName() == "rightBorder"))
            {
                (this->vx) = -(this->vx);
            }
            else
                if ((*pt2)->getName() == "upsideBorder")
                    (this->vy) = -(this->vy);
    }
```

Border

Pour les rebonds de la balle sur le border, le border ne va pas être cassé. Nous créons deux frontières pour éviter la situation que les variables **«vx»** et **«vy»** sont trop grandes que le program ne peut pas retenir les coups.

```
else {//rebondir la barre //les trois situations sur les parties de la barre
    if (vx = -1)
    {
        if (px == ((*pt2)->getPosition().x + static_cast <short>(3))) {
            (this->vx) = 0;
            (this->vy) = -(this->vy);
        }
        else {
            if (px < ((*pt2)->getPosition().x + static_cast <short>(3)))
            {
                (this->vy) = static_cast <short>(1);
            }
            else
            {
                (this->vx) = static_cast <short>(1);
                (this->vy) = static_cast <short>(1);
            }
        }
    }
    else
    {
        if (vx = 1) {
            if (px == ((*pt2)->getPosition().x + static_cast < short>(4))) {
                (this->vx) = 0;
                (this->vy) = -(this->vy);
            }
            else {
                if (px < ((*pt2)->getPosition().x + static_cast <short>(4)))
                {
                    (this->vy) = static_cast <short>(1);
                    (this->vx) = static_cast <short>(-1);
                }
                else
```

```
            {
                (this->vy) = static_cast <short>(1);
                (this->vx) = static_cast <short>(1);
            }
        }

}
else
{
    if (px == ((*pt2)->getPosition().x + 4)) {
        (this->vx) = 0;
        (this->vy) = -(this->vy);
    }
    else {
        if (px < ((*pt2)->getPosition().x + 4))
        {
            (this->vy) = 1;
            (this->vx) = -1;
        }
        else
        {
            (this->vx) = 1;
            (this->vy) = 1;
        }
    }
}
```

Barre

Si **«vx»**=1, le programme va comprarer le variable **«px»**, le largeur et la **position.x** de la barre pour juger la direction de la balle qui va repartir à gauche ou à droite ou l'opposée. Et si **«vx»**=-1 ou 0, la code de ce jugement est différent à cause de l'angle de la vue.

```
void Barre::update()
{

    if (GetAsyncKeyState(VK_RIGHT) && position.x < 50)
        position.x += 2;
    if (GetAsyncKeyState(VK_LEFT) && position.x > 5)
        position.x -= 2;
    Sleep(25);
}
```

* nous n'avons pas créé une classe dérivée de la classe **Console** et surcharger la méthode **«OnKeyPressed»**. Pour remplacer cette méthode, nous avons cherché les informations sur internet et utilisé une fonction **«GetAsynKeyState»** dans la méthode **«update»** de la barre pour contrôler le mouvement de la barre.

Le résultat du jeu

Échec :

```
if ((*pt2)->getName() != "abyss")

else {
    string sl = "You lose !!!";
    judge = 1;
    this->bodyManager->quit();
    this->bodyManager->getConsole().clear();
    this->bodyManager->getConsole().set_pen_color(red);
    this->bodyManager->getConsole().put_string(sl, 22, 20);

}
```

Nous définissons un objet **«sbyss»** de type Body, qui représente l'espace ci-dessous. Si la balle rebondit sur l'objet **«sbyss»**, le jeu est perdu. Chaque move et update le juge en premier lieu.
Nous utilisons la variable globale **«judge»** pour casser la brique qui se déplace. Si le jeu est perdu mais la brique qui se déplace n'est pas encore cassée, **"judge=1"**permet d'effacer la brique car la méthode **clear()** ne peut pas effacer la brique qui se déplace.
Si le jeu est perdu, on va effacer toutes les bodies sur l'écran et afficher "you lose!!!" .

Succès :

```
if ((this->numOfbriques) == 8) {
    string sw = "You win !!!";
    this->bodyManager->quit();
    this->bodyManager->getConsole().clear();
    this->bodyManager->getConsole().set_pen_color(green);
    this->bodyManager->getConsole().put_string(sw, 30, 20);
}
```

Nous avons 7 briques ensemble avec une d'eux qui doit recevoir 2 coups, donc si **«numOfbrique»**=8, toutes les briques sont cassés, donc on gagne le jeu.
Si le jeu est perdu, on va effacer toutes les bodies sur l' écran et afficher "you win!!!" .