There is no Planet B

MO LI, XIYING ZUO

Titre du jeu: les aventures d'Emily sur la planète

Introduction au jeu :

Emily vit sur terre, mais elle en a marre de la vie sur terre et veut voyager sur d'autres planètes. Pendant Noël, elle a eu deux semaines de vacances avec sept planètes (à l'exclusion de la terre), les huit planètes du système solaire. Chaque planète a ses propres conditions, et Emily doit utiliser les fonds initiaux pour acheter les conditions environnementales et de vie nécessaires pour survivre.

Les bases de la survie sur terre sont connues (elle a besoin de ces sept conditions pour survivre sur la planète):

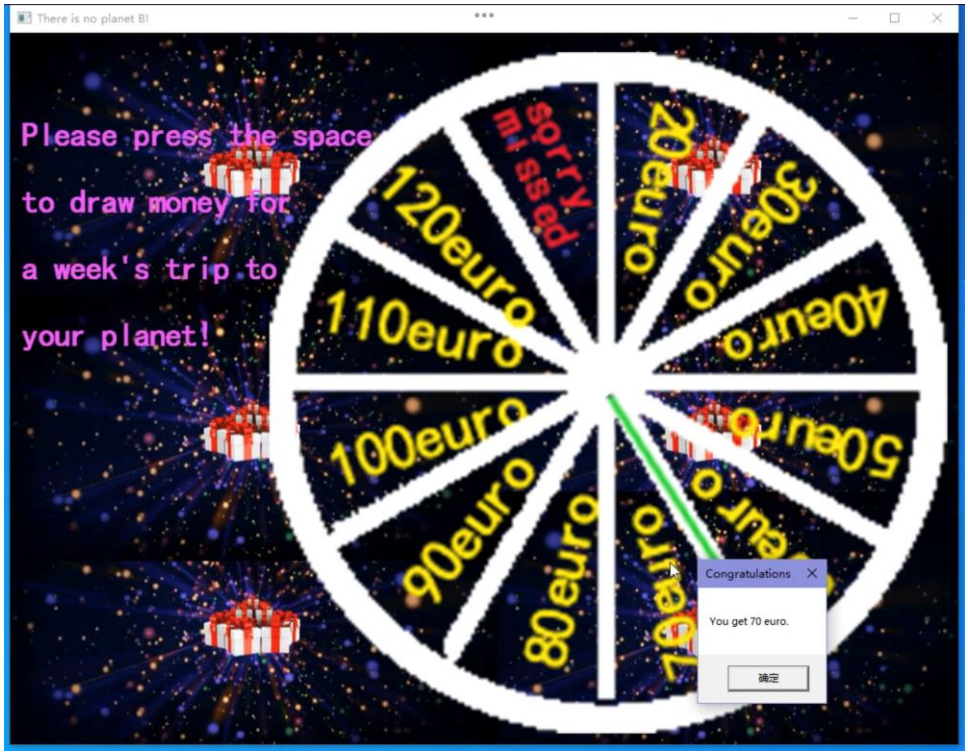
Environnement: soleil, température, oxygène, terre, gravité

Nécessités de la vie: nourriture, eau

Interface de jeu:

1. Interface initiale:

Emily tire des fonds de voyage à travers une table tournante, il peut tirer 0, 20, 30 euros... Jusqu'à 120 euros. Elle ne peut pas voyager et reprendre le jeu quand elle tire 0 €



2. Interface magasin：

Sachant que les sept planètes ont déjà les conditions de vie suivantes, Emily doit acheter dans les magasins ce qui manque aux planètes qu'elle veut aller,

Mercure: terre, soleil

Vénus: terre, soleil, gravité

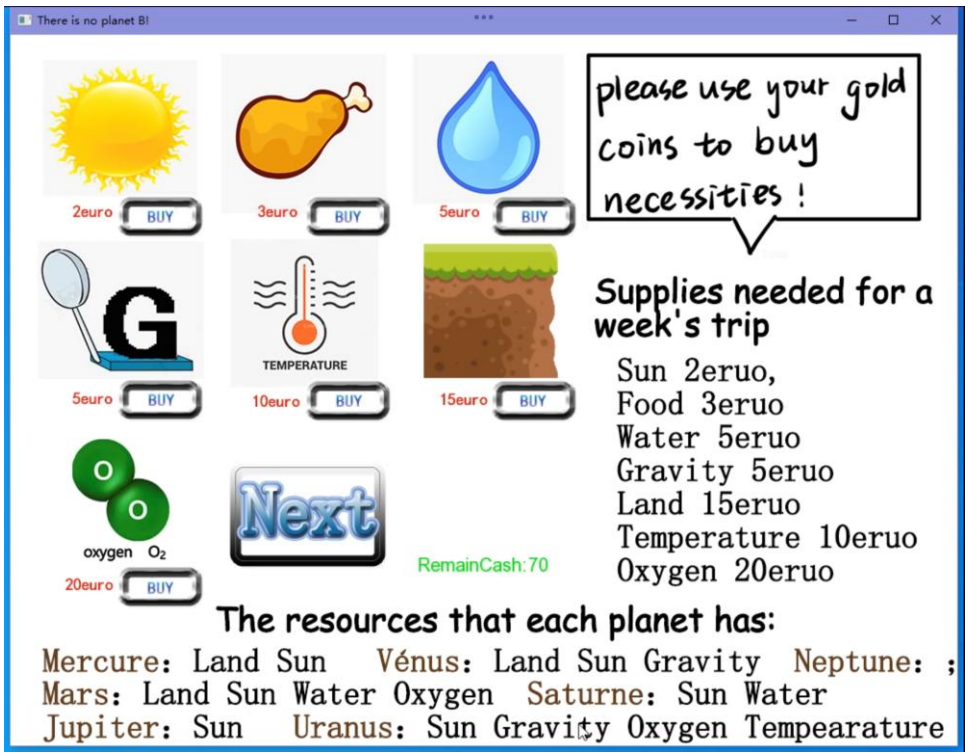
Mars: terre, soleil, eau, oxygène

Jupiter: Soleil

Saturne: soleil, eau

Uranus: soleil, gravité, oxygène, température

Neptune: rien



Après avoir cliqué sur le bouton d'achat, le solde correspondant sera réduit. Ensuite, cliquez sur le bouton suivant pour entrer dans l'interface select Planet

3. Interface select Planet

Nous avons sept planètes à choisir: mercure, mars, Jupiter, Saturne, Vénus, Neptune et Uranus.

Lorsque vous n'avez pas acheté le plein soleil, la température, l'oxygène, la terre, la gravité, la nourriture, l'eau, vous ne pouvez pas aller à la page suivante et vous serez rappelé ce qui manque pour voyager sur cette planète.

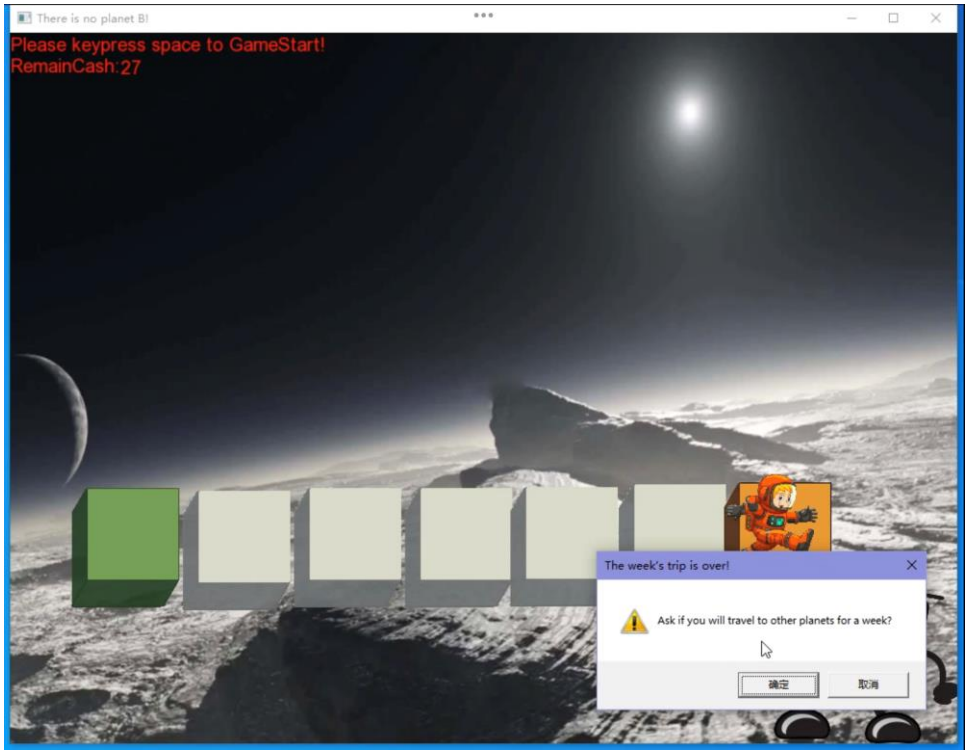
4. Interface à l'intérieur de la planète

Chaque planète a des grilles différentes, chacune représentant une journée, c'est - à - dire une semaine de voyage. Nous obtenons le nombre d'étapes que nous prenons en cliquant sur le dé aléatoire en bas à droite, qui coûte dix coins d'or par rouleau.

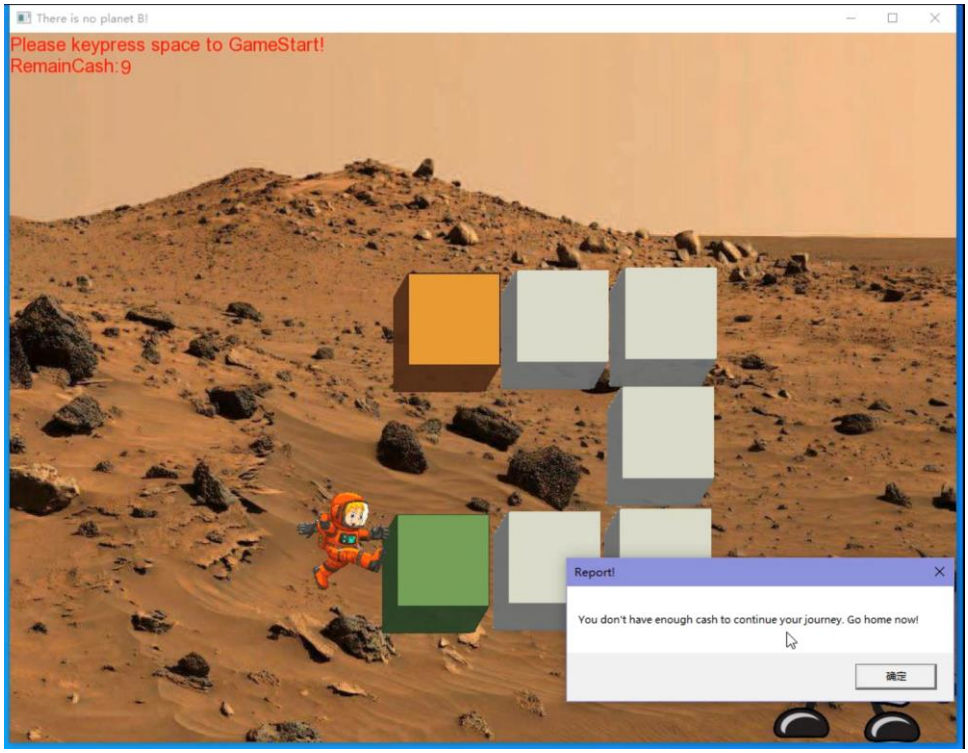
Si le coin change à 0 après le rouleau de dés, le jeu se retirera, et si vous avez encore de coins après sept carrés, vous serez invité à choisir de voyager une semaine de plus (vous retournerez à l'interface du magasin pour acheter les nécessités de la vie, répéter le jeu)

Ici nous avons choisi Uranus:

Ici, nous choisissons Mars comme deuxième voyage

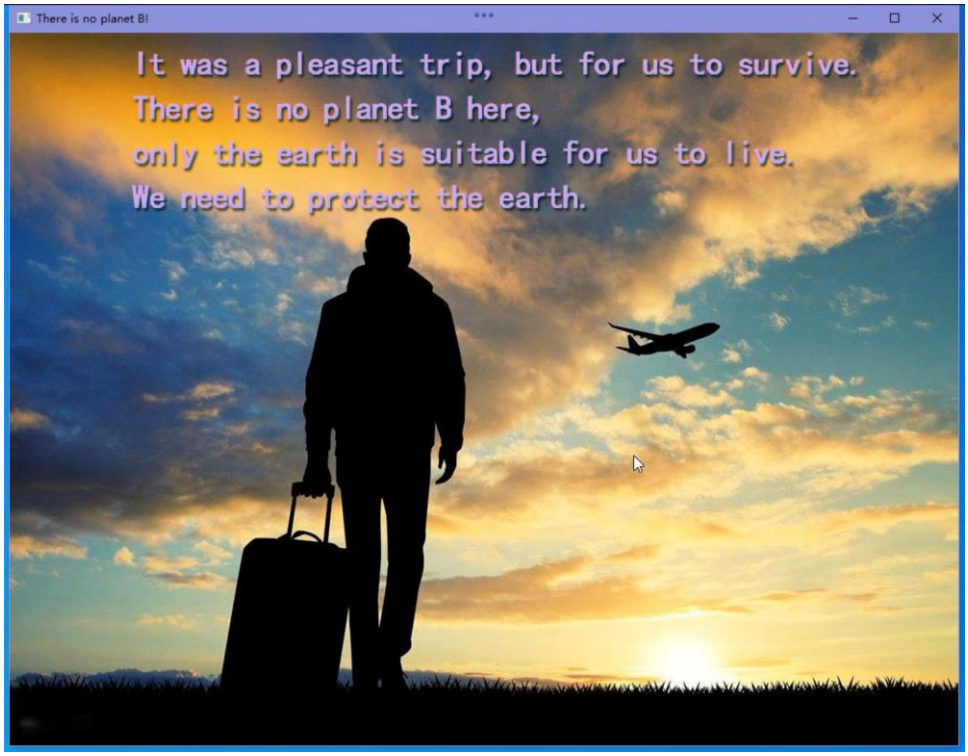


Ensuite, nous choisissons Mars comme deuxième voyage:



5. Interface de fin de jeu:

Emily termine son voyage sur la planète et retourne finalement sur terre. Bien que le voyage ait été agréable, il a été très court. La terre est l'endroit idéal pour vivre, nous devons la protéger parce qu'il n'y a pas de planète B.



- **Mise en œuvre du Code:**

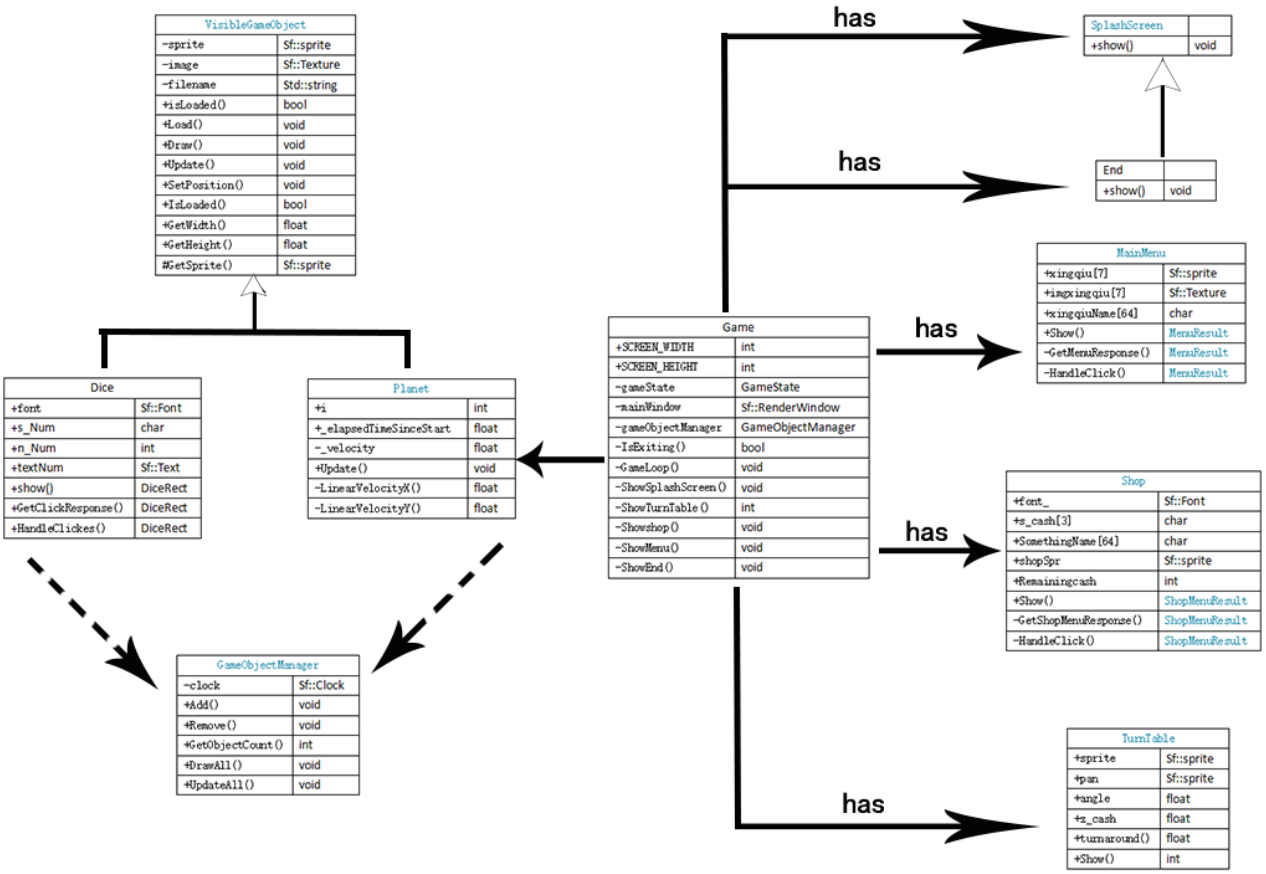
Mes huit classes sont： VisibleGameObject, Dice, Planet, GameObjectManager, Game, MainMenu, Shop, TumTable

Ce sont les 2 conteneurs différent:

```
std::map<std::string, VisibleGameObject*> _gameObjects;
```

```
std::list<MenuItem> _menuItems;
```

Diagramme UML:



- **La partie la plus fière:**

Dans le planet.cpp, nous avons mis en place sept grilles différentes sur chaque planète pour voyager.

```
void Planet::Update(float elapsedTime)
{
    _elapsedTimeSinceStart += elapsedTime;
    // Les gens marchent selon un certain temps
    if (_elapsedTimeSinceStart < 0.6f)
    {
        switch (i)
        {
        case 1:
        {
            switch (j)
            {
            case 0:   starPos = { -50,450 };GetSprite().setPosition(starPos); break;
            case 1:case 2:case 3:case 4:
                LinearVelocityX(9);break;
            case 5:
                LinearVelocityY(-9);break;
            case 6:case 7:
                LinearVelocityX(-9);break;
            }
            break;
        default:

            break;
```

```
case 2:
    switch (j)
    {
    case 0:   starPos={ 303,535 };GetSprite().setPosition(starPos); break;
    case 1:case 2:case 3:
        LinearVelocityX(9);break;
    case 4:case 5:
        LinearVelocityY(-9);break;
    case 6:case 7:
        LinearVelocityX(-9);break;
    default:
        break;
    }
    break;
```

```cpp
float Planet::LinearVelocityX(int x)
{
    px=GetSprite().getPosition().x;
    px+=x;
    sf::sleep(sf::milliseconds(30));
    if ( px > 1020)
    {
        px = 1020;
    }
    if( px < -60)
    {
        px =-60;
    }
    GetSprite().setPosition(px,GetSprite().getPosition().y);
    return GetSprite().getPosition().x;
}
float Planet::LinearVelocityY(int y)
{
    py=GetSprite().getPosition().y;
    py+=y;
    sf::sleep(sf::milliseconds(30));
    if(py>760)
        py=760;
    if(py<10)
        py=10;
    GetSprite().setPosition(GetSprite().getPosition().x,py);
    return GetSprite().getPosition().y;
}
```

- **Exécution du Code:**

J'utilise Visual Studio sous Windows. Après l'exécution, j'a i généré un fichier exe. Vous pouvez démarrer le jeu directement en cliquant dessus. (pour éviter que votre système ne soit incompatible, j'ai téléchargé une petite vidéo de mon jeu sur le site github)