打飞机工程实验报告

——By 杨晨 2014060201016

1. 给谁做的？

给那些一天工作或学习下来想要放松下心情的人做的，让他们缓解压力。

2. 做什么？也就是有哪些功能？ 游戏有哪些规则？

2.1. 玩家有几条命？

3 条命。

2.2. 如何开炮？

按空格键。

2.3. 如何移动？

键盘左右键。

2.4. 敌机怎么出来？

随机出来， 按一定时间间隔。

2.5. 敌机飞行的路径、速度？

向下， 随着关卡变快。

2.6. 有没有礼包？

还没加入。

2.7. 有没有关卡？

有。

2.8. 游戏速度如何变化。

线性变化。

3. 怎么做？

3.1. 使用哪些工具？

CodeBlocks

3.2. 库？

SFML

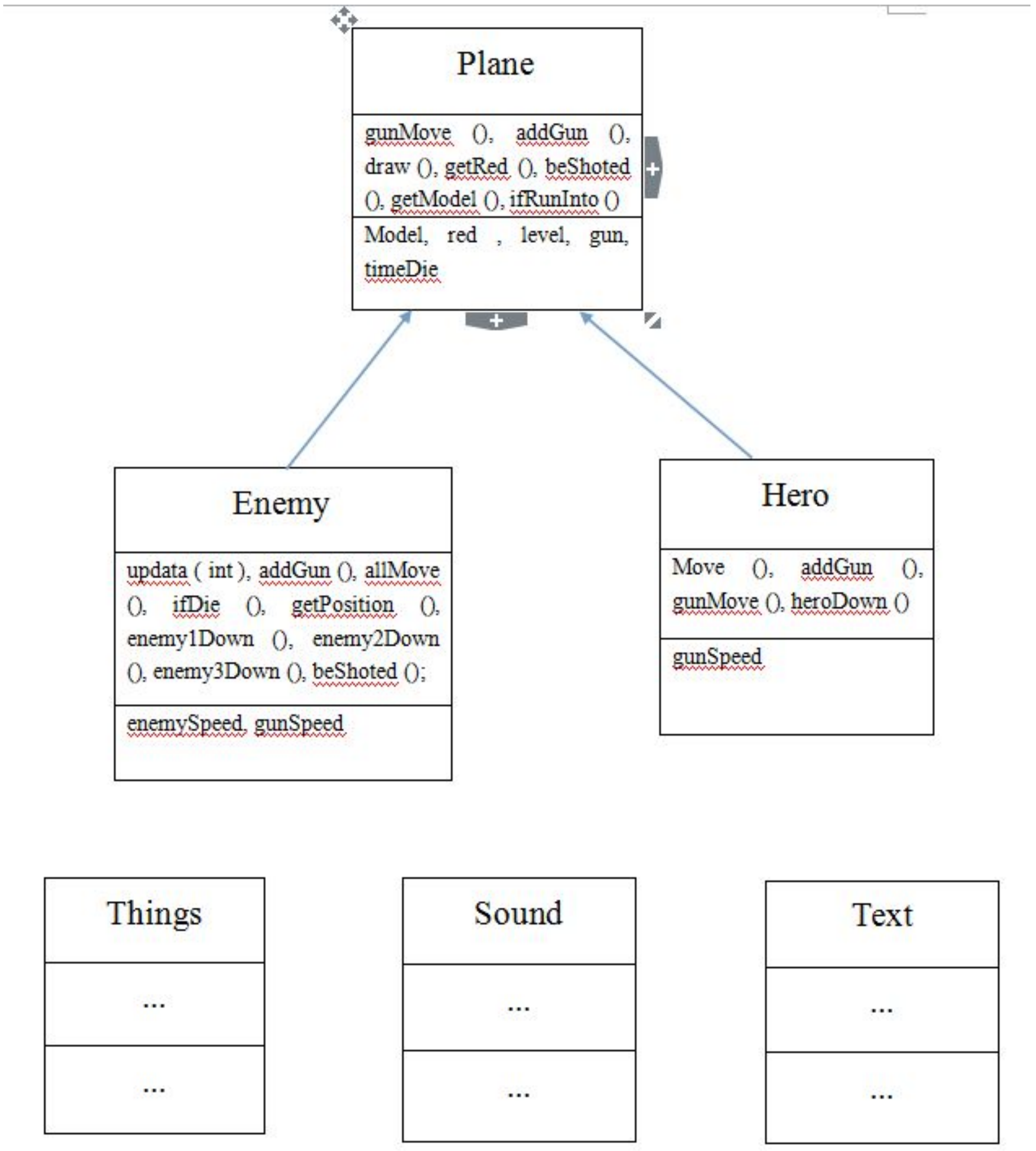
3.3. 如何配置开发环境？

详见：http://www.sfml-dev.org/tutorials/2.3/start-cb.php

3.4. 实战：模块（一般以类为单位）如何划分？

Plane 一个类， Enemy 和 Hero 继承 Plane。 Things， Text， Sound 为 3 个静态写的类， 放的是一些预处理内容， 让其他地方直接调用。

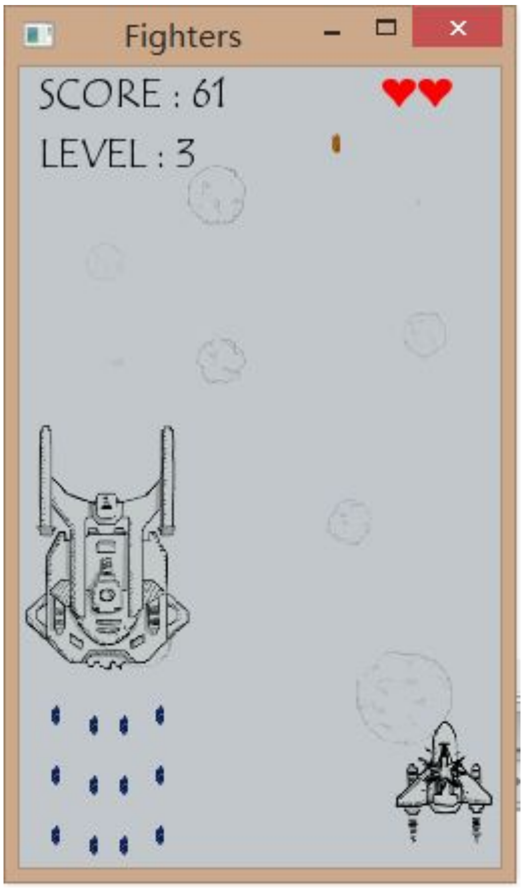
3.5. 关系如何（画 UML 表达）？


Plane
gunMove (), addGun (), draw (), getRed (), beShoted (), getModel (), ifRunInto ()
Model, red , level, gun, timeDie
Enemy
updata (int), addGun (), allMove (), ifDie (), getPosition (), enemy1Down (), enemy2Down (), enemy3Down (), beShoted ();
enemySpeed, gunSpeed
Hero
Move (), addGun (), gunMove (), heroDown ()
gunSpeed
Things
...
...
Sound
...
...
Text
...
...

（ 因为代码实在不好贴，所以在后面贴吧。还有因为 Things， Sound，Text。里面都是些静态成员所以没有写， 并且与主要的 Plane 并没有继承关系。 ⊙﹏⊙b 汗 ）

4. 做得如何？

4.1. 运行时的界面截图



4.2. 程序运行速度

没有卡顿， 挺流畅。

4.3. 程序大小

900 行左右。

4.4. 玩家体验、评价

挺有意思， 再加些功能因该更有意思。

5. 其他

Plane 类：

```
class Plane
{
    public:
        Plane ( sf::Texture&, int);
        void gunMove ( int );
        void addGun ( sf::Texture& );
        void draw ( sf::RenderWindow& );
        int getRed ();
        void beShoted ();
        sf::Sprite& getModel ();
        bool ifRunInto ( sf::Sprite& );
        virtual ~Plane();
    protected:
        sf::Sprite model;
        int red, level;
        LIST_GUN gun;
        int timeDie;
};

Plane::Plane ( sf::Texture& thing, int lv )
{
    model.setTexture ( thing );
    model.setScale (sf::Vector2f(0.5f, 0.5f));
    level = lv;
    if ( lv == 0 )
    {
        model.setPosition (sf::Vector2f( 180*0.5, 650*0.5));
        red = 3;
    }
    if ( lv == 1 )
    {
        int x = rand ()%200;
        model.setPosition (sf::Vector2f( x, 10));
        red = 1;
    }
    if ( lv == 2 )
```

```
		{
			int x = rand ()%200;
			model.setPosition (sf::Vector2f( x, 10));
			red = 2;
		}
		if ( lv == 3 )
		{
			int x = rand ()%150;
			model.setPosition (sf::Vector2f( x, 0));
			red = 3;
			Sound::ENEMY3_FLY.play ();
		}

		timeDie = 0;
	}

	void Plane::gunMove ( int v )
	{
		LIST_GUN::iterator i;
		for ( i=gun.begin(); i!=gun.end(); i++ )
		{
			(*i).move (0, v);
		}

		for ( i=gun.begin(); i!=gun.end(); )
		{
			sf::Vector2f pos = (*i).getPosition();
			if ( pos.y<0 || pos.y>400 )
			{
				gun.erase ( i++ );
			}
			else i++;
		}
	}

	void Plane::addGun ( sf::Texture& thing )
	{
		sf::Sprite temp;
		sf::Vector2f pos = model.getPosition();
		temp.setTexture ( thing );
		temp.setScale ( sf::Vector2f(0.5f, 0.5f) );
		if ( level < 2 )
		{
			if ( level == 0)
```

```
                {
                        pos.y -= 12;
                        pos.x += 24;
                }
                if ( level == 1)
                {
                        pos.x += 13;
                        pos.y += 23;
                }
                temp.setPosition ( sf::Vector2f ( pos.x, pos.y) );
                gun.push_back (temp);
        }
        if ( level == 3)
        {
                pos.x += 15;
                pos.y += 125;
                temp.setPosition ( sf::Vector2f ( pos.x, pos.y) );
                gun.push_back (temp);
                pos.x += 85-33;
                temp.setPosition ( sf::Vector2f ( pos.x, pos.y) );
                gun.push_back (temp);
                pos.y += 5;
                pos.x += 32-85+20;
                temp.setPosition ( sf::Vector2f ( pos.x, pos.y) );
                gun.push_back (temp);
                pos.x += 15;
                temp.setPosition ( sf::Vector2f ( pos.x, pos.y) );
                gun.push_back (temp);
        }
        if ( level == 2)
        {
                pos.x += 8;
                pos.y += 50;
                temp.setPosition ( sf::Vector2f ( pos.x, pos.y) );
                gun.push_back (temp);
                pos.x += 16;
                temp.setPosition ( sf::Vector2f ( pos.x, pos.y) );
                gun.push_back (temp);
        }
}

void Plane::draw ( sf::RenderWindow& window )
{
        window.draw (model);
        LIST_GUN::iterator i;
        for ( i=gun.begin(); i!=gun.end(); i++ )
        {
            window.draw ( (*i) );
        }
    }

    int Plane::getRed ()
    {
        return red;
    }

    sf::Sprite& Plane::getModel ()
    {
        return model;
    }

    bool Plane::ifRunInto ( sf::Sprite& thing )
    {
        LIST_GUN::iterator i;
        for ( i=gun.begin(); i!=gun.end(); i++ )
        {
            if ((*i).getGlobalBounds().intersects
(thing.getGlobalBounds()))
            {
                gun.erase (i++);
                return true;
            }
        }
        return false;
    }

    void Plane::beShoted ()
    {
        red--;
    }
```

Enemy 类：

```
    class Enemy : public Plane
    {
        public:
            Enemy ( sf::Texture&, int, int );
        void updata ( int );
        void addGun ();
        void allMove ();
        bool ifDie ();
        sf::Vector2f getPosition ();
        void enemy1Down ();
        void enemy2Down ();
        void enemy3Down ();
        void beShoted ();
        virtual ~Enemy();
    protected:
    private:
        float enemySpeed, gunSpeed;
};

Enemy::Enemy ( sf::Texture& thing , int lv, int pass ) : Plane ( thing, lv )
{
    updata ( pass );
}

void Enemy::updata ( int pass )
{
    enemySpeed = 1+pass*0.5;
    gunSpeed = 2.5+pass*0.5;
}

void Enemy::addGun ()
{
    Plane::addGun ( Things::BUTTON );
    Sound::BUTTON.play ();
}

void Enemy::allMove ()
{
    if ( red > 0 )
        model.move ( 0, enemySpeed );
    else
    {
        if ( 5==timeDie )
        {
            red--;
            if ( 1==level )enemy1Down ();
            if ( 2==level )enemy2Down ();
            if ( 3==level )enemy3Down ();
```

```
                timeDie = 0;
            }
            timeDie++;
        }
        Plane::gunMove ( gunSpeed );
}

sf::Vector2f Enemy::getPosition ()
{
        return model.getPosition ();
}

bool Enemy::ifDie ()
{
        if ( 3 > level )
            if ( red < -3 && gun.empty () )
                return true;
        if ( 3 ==level )
            if ( red < -5 && gun.empty () )
                return true;
        return false;
}

void Enemy::enemy1Down ()
{
        if ( red == 0 )
            model.setTexture ( Things::ENEMY1_DOWN1 );
        if ( red == -1 )
            model.setTexture ( Things::ENEMY1_DOWN2 );
        if ( red == -2 )
            model.setTexture ( Things::ENEMY1_DOWN3 );
        if ( red == -3 )
            model.setTexture ( Things::ENEMY1_DOWN4 );
        if ( red == -4)
            model.setTexture ( Things::EMPTY );
}

void Enemy::enemy2Down ()
{
        if ( red == 1 )
            model.setTexture ( Things::ENEMY2_HIT );
        if ( red == 0 )
            model.setTexture ( Things::ENEMY2_DOWN1 );
        if ( red == -1 )
            model.setTexture ( Things::ENEMY2_DOWN2 );
        if ( red == -2 )
            model.setTexture ( Things::ENEMY2_DOWN3 );
        if ( red == -3 )
            model.setTexture ( Things::ENEMY2_DOWN4 );
        if ( red == -4)
            model.setTexture ( Things::EMPTY );
}

void Enemy::enemy3Down ()
{
        if ( red == 2 )
            model.setTexture ( Things::ENEMY3_HIT1 );
        if ( red == 1 )
            model.setTexture ( Things::ENEMY3_HIT2 );
        if ( red == 0 )
            model.setTexture ( Things::ENEMY3_DOWN1 );
        if ( red == -1 )
            model.setTexture ( Things::ENEMY3_DOWN2 );
        if ( red == -2 )
            model.setTexture ( Things::ENEMY3_DOWN3 );
        if ( red == -3 )
            model.setTexture ( Things::ENEMY3_DOWN4 );
        if ( red == -4 )
            model.setTexture ( Things::ENEMY3_DOWN5 );
        if ( red == -5 )
            model.setTexture ( Things::ENEMY3_DOWN6 );
        if ( red == -6)
            model.setTexture ( Things::EMPTY );
}

void Enemy::beShoted ()
{
        Plane::beShoted ();
        if ( red==0 )
        {
            Text::ans++;
            if ( 1==level )
            {
                Sound::ENEMY1_DOWN.play();
            }
            if ( 2==level )
            {
                Sound::ENEMY2_DOWN.play();
```

```
                Text::ans++;
            }
            if ( 3==level )
            {
                Sound::ENEMY3_DOWN.play();
                Text::ans += 4;
            }
        }
        else
        {
            if ( level == 2 )
                enemy2Down ();
            if ( level == 3 )
                enemy3Down ();
        }
    }
```

Hero 类：

```
class Hero: public Plane
{
    public:
        Hero( void );

        void move ( int );
        void addGun ();
        void gunMove ();
        void heroDown ();
        virtual ~Hero();
    protected:
    private:
        int gunSpeed;
};

Hero::Hero( void ) : Plane ( Things::HERO, 0)
{
    gunSpeed = -5;
}

void Hero::move ( int x )
{
    Plane::model.move ( x, 0);
    sf::Vector2f pos = model.getPosition();
    if (pos.x<=0) model.move(3, 0);
    if (pos.x>=240-53) model.move(-3, 0);

}

void Hero::addGun ()
{
    Plane::addGun ( Things::BULLET );
    Sound::BULLET.play ();
}

void Hero::gunMove ()
{
    if ( getRed()<1 )
    {
        if ( 10==timeDie )
        {
            red--;
            timeDie = 0;
            heroDown ();
        }
        else timeDie++;
    }
    Plane::gunMove ( gunSpeed );
}

void Hero::heroDown ()
{
    if ( red == 2 )
        model.setTexture ( Things::HERO_BLOWUP1 );
    if ( red == 1 )
        model.setTexture ( Things::HERO_BLOWUP2 );
    if ( red == 0 )
        model.setTexture ( Things::HERO_BLOWUP3 );
    if ( red == -1 )
        model.setTexture ( Things::HERO_BLOWUP4 );
}
```

最后 谢谢观看 O(∩_∩)O~