# 94 组：Qt 大作业报告

组名：你说得队　组长：吴童 组员：商邑飞、魏甬翔

0.程序名称：信科教吾物语之宗熹历险记

1.程序功能介绍：

1.1 玩家操纵英雄在预设地图上的平滑转向和连贯行走

1.2 玩家与怪物的回合制对战

1.2.1 怪物基于地牢场景的随机生成（包括类型、数量和位置）

1.2.2 怪物与玩家的回合制对战结算

1.3 家园、村庄和地牢三大场景间的感应切换

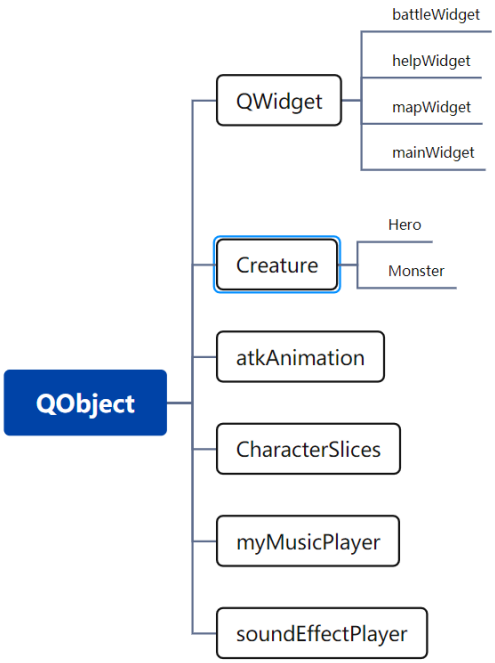
1.4 帮助界面、英雄信息的键盘呼出

1.5 自动存档和读档

1.6 英雄各种行为对应动画和特效的设计与展示

2.项目各模块与类设计细节：

2.0 对象树




2.1 主界面

```
class mainWidget : public QWidget

{
    Q_OBJECT

public:
    mainWidget(QWidget *parent = nullptr);
    void startGame ();
    ~mainWidget();

protected:
    void keyPressEvent(QKeyEvent *event);

private slots:
    void on_endButton_clicked();

    void on_startButton_clicked();

    void on_helpButton_clicked();

private:
    Ui::wuyu *ui;
};
```

2.1.1　三个信号和槽机制，分别支持鼠标点击启动/退出游戏和打开菜单

2.1.2　QKeyEvent 类，支持键盘 Esc 退出游戏，Enter 开始游戏，H 呼出菜单

```
    myMusicPlayer::stopMusic();
    this->close();
//    mciSendString(TEXT("close S1"),NULL,0,NULL);
    Hero* hero = new Hero();
    std::ifstream is("../wuyu/data/hero/hero.txt");
    std::string lastMap,_name;
    is>>lastMap>>hero->maxHp>>hero->hp>>hero->atk>>hero->def>>hero->mag>>_name;
    hero->name = QString::fromStdString(_name);
//    qDebug() << "Hero created successfully!";
    mapWidget* m = new mapWidget(nullptr ,hero, this);
    m->load(QString::fromStdString(lastMap));
```

2.1.3　启动游戏，载入英雄对象、地图信息（自动读档与存档，包括英雄的出生位置、朝向和英雄数据）、背景音乐和背景淡入淡出特效

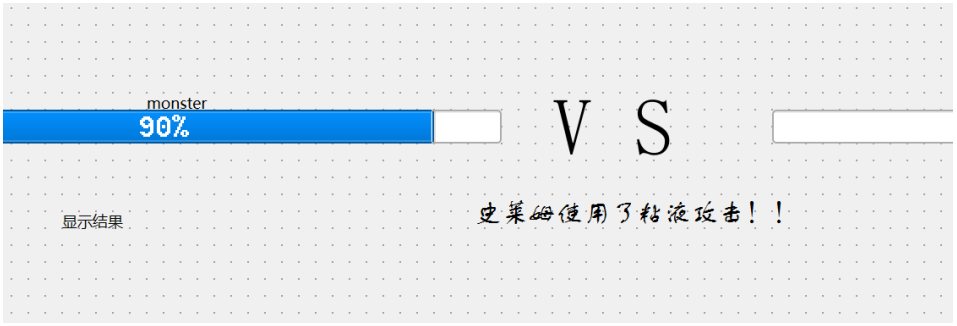
2.2　帮助界面


2.2.1　该界面主体是一个 UI 界面设计，设置了数个 QLabel 控件提供游戏相关按键索引提示和版本信息，加入界面对齐和弹簧控件优化排版

```
class helpWidget : public QWidget
{
    Q_OBJECT

public:
    explicit helpWidget(QWidget *parent = nullptr);
    ~helpWidget();

protected:
    void keyPressEvent(QKeyEvent *event);

private:
    Ui::helpWidget *ui;
};
```

2.2.2 信号和槽机制配合主菜单实现帮助界面的呼入和呼出

## 2.3 战斗界面





```
srand(time(0));
switch (rand()%2) {
    case 0:
        myMusicPlayer::playMusic("battle1", 30);
        break;
    case 1:
        myMusicPlayer::playMusic("battle2", 30);
        break;
}
```

2.3.1 UI 界面：QPushButton 控件通过信号和槽关联英雄回合内的行动，QLabel 控件展示行动效果和体力结算，QProgressBar 将英雄和怪物的 hp 值可视化。在构造函数中载入 UI 界面的初始化、英雄指针和怪物指针（怪物有两类），根据时间进行回合的交替

```
private slots:
    void on_baseAttackButton_clicked();
    void monsterAttack();
    void on_fleedButton_clicked();
    void on_skill1Button_clicked();
    void on_skill2Button_clicked();
    void on_skill3Button_clicked();

protected:
    void keyPressEvent(QKeyEvent *event);
```

2.3.2 信号和槽机制关联英雄和怪物回合内的行动：普通攻击、炎爆术、初级

治疗术、空过（方便调试）、逃跑（方便调试）、怪物自动普通攻击（按钮调试用）；相应键盘快捷键：退格键逃跑（方便调试），其余不赘述。技能的触发以及体力结算伴随音效和动画

```
void battleWidget::maintenance() {      //维护血条和胜负判定
    //qDebug() <<"Function maintenance not completed!";
    if (isGameover) return;
    isWorking = 1;
    //hero->PrintInfo();
    qDebug() << hero->hp << monster->hp;
    if (hero->hp == 0) {
        isGameover = 1;
        ui->resultLabel->resize(this->size());
        ui->resultLabel->move(0,0);
        ui->resultLabel->setPixmap(QPixmap("../wuyu/images/battle/defeat.png"));
        ui->resultLabel->raise();
        ui->resultLabel->show();
        myMusicPlayer::playMusic("defeat");
        CharacterSlices::delay(3000);

        std::fstream is("../wuyu/data/mapWidget/dungeon0.txt", std::ios::in);
        std::fstream os("../wuyu/data/mapWidget/dungeon.txt", std::ios::out | std::ios::trunc);
        while (!is.eof()) {
            char str[1024];
            is.getline(str,sizeof(str),'\n'); //读一行
            os << str <<std::endl; //输出到文件
        }
        is.close();
        os.close();

        isWorking = 0;
        hero->Revive();
        map->load("village");
        returnToMap();
    }
    if (monster->hp == 0) {
        isGameover = 1;
        ui->resultLabel->resize(this->size());
        ui->resultLabel->move(0,0);
        ui->resultLabel->setPixmap(QPixmap("../wuyu/images/battle/victory.png"));
        ui->resultLabel->raise();
        ui->resultLabel->show();
        myMusicPlayer::playMusic("victory");
        CharacterSlices::delay(3000);

        isWorking = 0;
        map->killMonster();
        returnToMap();
    }
```

2.3.3　每一回合后的结算过程，同时维护 UI 界面和自动读存档，进行胜负判定并弹出相应动画和音效。以下是具体的维护过程：根据我方攻击力、技能倍率、敌方防御三个数值通过结算公式得到不同的伤害值，分别反应到角色/怪物上方的文字实时行动信息和进度条控件上，并通过 log 实时显示玩家和怪物的行动信息。胜负判定成真后会回到地图同时触发相应的战斗结果画面和音效。

## 2.4　动画与音效

```cpp
soundEffectPalyer::soundEffectPalyer()
{
    this->audioOutput = new QAudioOutput;
    this->mediaPlayer = new QMediaPlayer;
    mediaPlayer->setLoops(1);
}
soundEffectPalyer::~soundEffectPalyer()
{
    delete this->audioOutput;
    delete this->mediaPlayer;
}

void soundEffectPalyer::playMusic(QString musicName, int volume)
{
    //mediaPlayer->stop();
    if (musicName != lastEffect)
    {
        mediaPlayer->setSource(QUrl::fromLocalFile("../wuyu/music/" + musicName + ".mp3"));
        lastEffect = musicName;
    }
    audioOutput->setVolume(volume);
    // qDebug() <<  mediaPlayer->hasAudio() <<mediaPlayer->source();
    mediaPlayer->setAudioOutput(audioOutput);
    mediaPlayer->play();
}
void soundEffectPalyer::stopMusic()
{
    mediaPlayer->stop();
}
```

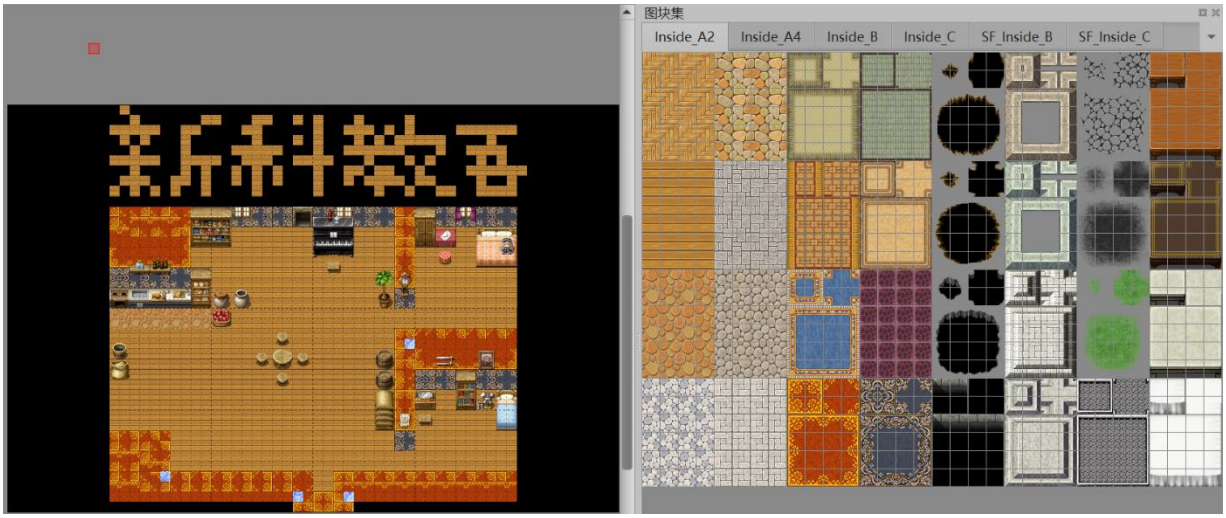
2.4.1 音效：英雄在各地图行进、战斗时和战斗胜利后通过 Qtmultimdia 库播放相关音效。特别地，在战斗时，背景音乐和技能特效同时具备音效，故在实践层面我们采用了两个 QObeject 类实现并行操作。

2.4.2 行进动画与技能特效：我们的处理方法是先采集一张任务各个方向/技能各个阶段的底图，将其切割成一串子图后导入 QPixmap 对象数组，使用时在同一 Label 上依次播放，考虑到播放效果，我们对播放过程进行了延时处理

```cpp
CharacterSlices::CharacterSlices(const QString & picAddress, int picID,
                                 const QString & faceAddress)
{
    //初始化face
    QPixmap faceMap(faceAddress);
    if (faceMap.isNull())qDebug() << "error! null pixmap";
    int x = (picID-1)/2;
    int y = (picID+1)%2;
    QRect facePosition(QPoint(x*CHARATER_FACE_SIZE, y*CHARATER_FACE_SIZE),
                       QPoint((x+1)*CHARATER_FACE_SIZE, (y+1)*CHARATER_FACE_SIZE) );
    face = faceMap.copy(facePosition).scaled(QSize(FACE_APPLY_SIZE, FACE_APPLY_SIZE),
                                             Qt::KeepAspectRatio);

    int ptr = 0;
    QPixmap map(picAddress);

    int startW, startH;
    if (picID%2)
    {
        startW = (picID/2)*3;
        startH = 0;
    }
    else
    {
        startW = (picID/2 - 1)*3;
        startH = 4;
    }

    for (int i = startW; i < startW + 3; ++i)
        for (int j = startH; j < startH + 4; ++j)
        {
            int width = CHARATER_PIC_WIDTH;
            int length = CHARATER_PIC_LENGTH;
            //qDebug() << i << ' ' << j << " i and j len";
            QRect posi(QPoint(i*width, j*length), QPoint(i*width+width, j*length+length));
            slices[ptr++] = map.copy(posi);
            QSize picSize(CHARATER_APPLY_SIZE, CHARATER_APPLY_SIZE);
            slices[ptr-1] = slices[ptr-1].scaled(picSize, Qt::KeepAspectRatio);
        }
}
void CharacterSlices::delay(unsigned long int delayTime)
{
    QTime dieTime = QTime::currentTime().addMSecs(delayTime);
    while (QTime::currentTime() < dieTime)
        QCoreApplication::processEvents(QEventLoop::AllEvents, 10);
}
```

## 2.5 地图

2.5.1 地图的底图采用 Tiles 软件进行不同风格元素的标准方格铺排，通过底层信息矩阵数组（标注地图的状态：不可通行、可通行、可传送）的支持，角色可在地图部分位置自由移动，具体到村庄是黄色土路、家园是实木地板、地牢是卡其色土块和梯子，在有方向键指引的方格处，角色将完成地图间的传送。



```
if (direction == "left")
{
    posX--;
    ptr = charaPic.standFaceLeft;
    delta = QPoint(-CHARATER_STEP_LENGTH, 0);
}
else if(direction == "right")
{
    posX++;
    ptr = charaPic.standFaceRight;
    delta = QPoint(CHARATER_STEP_LENGTH, 0);
}
else if (direction == "up")
{
    posY--;
    ptr = charaPic.standFaceUp;
    delta = QPoint(0, -CHARATER_STEP_LENGTH);
}
else if (direction == "down")
{
    posY++;
    ptr = charaPic.standFaceDown;
    delta = QPoint(0, CHARATER_STEP_LENGTH);
}
else
{
    //qDebug() << "Wrong direction!";//如果方向输入错误，直接返回，debuglog
    return;
}
dir = ptr;
for (int i = 0; i < GRID_SIZE/CHARATER_STEP_LENGTH; ++i)
{
    ptr += 4;//距离下一个差分是4
    ptr = ptr % 12;//一共12个差分，防止下标越界
    lb.setPixmap(charaPic.slices[ptr]);
    position += delta;//位置向指定方向移动一步
    lb.move(position);//移动标签
    lb.show();
    charaPic.delay();//delay一下，以便看清动画
}
lb.setPixmap(charaPic.slices[dir]);
isWalking = false;

//与怪物的encounter
if (monsterMap[posX][posY] != nullptr)
{
    battleWidget* bw = new battleWidget(nullptr, hero, monsterMap[posX][posY]);
    bw->setMap(this);
    //this->save();//不保存，直接load村庄即可
    this->hide();
    bw->show();
}
```

2.5.2 角色在地图上的行走（包含可行性判定和边界判定）与传送以及配合相关按钮和热键的信号与槽机制

```
void mapWidget::generateMonsters (int num) {
    int cnt = 0;
    srand(time(0));
    CharacterSlices slime("../wuyu/images/character/Monster.png", 3);
    CharacterSlices bat("../wuyu/images/character/Monster.png", 1);
//    QLabel* testlb = new QLabel(this);
//    testlb->setPixmap(slime.slices[slime.standFaceDown]);
//    testlb->resize(CHARATER_APPLY_SIZE, CHARATER_APPLY_SIZE);
//    testlb->move(32, 32);
    while (cnt < num) {
        for (int i = 0; i < 30; ++i)
            for (int j = 0; j < 20; ++j)
            {
                if (cnt >= num)break;
                if (pass[i][j] != ORDINARY_GRID)
                    continue;
                if (std::abs(i-posX) <= 2 && std::abs(j-posY) <= 2)
                    continue;
                if (monsterMap[i][j] != nullptr)
                    continue;
                if (rand()%20 == 0)
                {
                    QLabel* t = new QLabel(this);
                    if (rand()%2)
                    {
                        monsterMap[i][j] = new Monster(80,20,5,4,"Slime",t);
                        t->setPixmap(slime.slices[slime.standFaceDown + rand()%4]);
                    }
                    else
                    {
                        monsterMap[i][j] = new Monster(60,40,0,5,"Bat",t);
                        t->setPixmap(bat.slices[bat.standFaceDown + rand()%4]);
                    }
                    t->resize(CHARATER_APPLY_SIZE, CHARATER_APPLY_SIZE);
                    t->move(i*GRID_SIZE-2, j*GRID_SIZE-2);
                    t->lower();
                    t->show();
                    ++cnt;
                }
            }
    }//while
}
```

2.5.3　地牢的怪物生成机制，包括怪物数量、类型、朝向和出生地的随机生成逻辑（目前仅支持 Slime 和 Bat 两种怪物）

2.6 其它不能在最终程序中展示的类

```
int maxHp;
int hp;
int atk;
int def;
int mag;
QString name;
QLabel* tachie;//立绘的日语罗马音,对战斗中的立绘进行操作

Creature(int, int, int, int, QString, QLabel* tachi = nullptr);
//Creature & operator = (const Creature & creat);
void baseAttack(Creature & opponent);
void labelShake(int radius = 20);//radius是立绘晃动的半径
void labelShakeMirror(int radius = 20);//上一个函数进行镜像翻转
virtual ~Creature();    //虚析构函数
void Revive();
//virtual void DefaultAttack(Creature* pEnemy) = 0;

int DealDamage (double rate, Creature& enemy);   //伤害结算函数
int DealRecovery (double rate);
```

2.6.1　以英雄类为例，该类包含了英雄的血量、攻击、防御、名称、立绘等各类详细信息

3.小组成员分工情况：

界面：

| | |
|---|---|
| 主界面 | 吴童 |
| 帮助界面 | 魏甬翔 |
| 地图绘制 | 吴童&商邑飞 |
| 战斗界面&结算 | 吴童&商邑飞 |
| 英雄信息界面 | 吴童 |

动画与特效：

| | |
|---|---|
| 英雄行走动画 | 商邑飞 |
| 技能特效 | 魏甬翔 |
| 战斗动画 | 吴童 |

其他：

| | |
|---|---|
| 音乐 | 吴童&商邑飞 |
| 屏幕淡入淡出 | 商邑飞 |
| 相关报告制作 | 魏甬翔 |
| 素材收集整理 | 商邑飞 |

4.项目总结与反思

4.1 基于 Qt 的 C++开发的相关感悟：

Qt 是一个跨平台的集成开发环境，合作期间，我们尝试了跨越 Windows 和 Mac 的共同开发，大大减少了开发和维护不同平台版本的工作量。Qt 提供了丰富的 UI 控件和开发工具，在面向对象的 C++开发的基础上，我们尝试可视化地构建整洁、优雅的交互界面，将更多平凡而冗长的设定代码封装在可视的控件之下。Qt 支持别具一格

的信号-槽机制，我们借助其实现不同事件之间的松耦合，使得事件与事件、事件与控件间有序关联同时边界独立，调试、剥离和重组也更为轻松。Qt 本身拥有丰富的功能库，我们借助 QtMultimedia 类和 QPixmap 类优化了本游戏在视觉和听觉上的体验。此外，Qt 提供了翔实的帮助文档、示例代码和视频教程，为我们的开发提供了不少便利。

4.2 基于合作开发的相关感悟：

4.2.1 时间和进度管理：为了开发该游戏，我们经历了立项-框架设计-分工-整合-再分工-再整合-总结的全过程。其中，随着环节的推进，我们耗费的时间不断下降。为此，我们认识到一个好的创意是一个项目的生命，而提前确立框架能够大大减少分工的纠缠性。为了确保进度符合预期，为每一个环节设置 deadline 是重中之重。

4.2.2 资源管理：我们尝试了基于微信群的定时汇总，在开发的过程中不断设置和更新锚点，追踪进度的同时以防最新功能的添加造成整个项目的巨大波动和紊乱。


4.2.3 团队合作与沟通：我们的沟通主要分为通过线上微信群和个人微信交换推进细节和通过线下例会头脑风暴整体设计和基本架构两个方面。虽然在立项之初有所蹉跎，但总的成效符合团队预期。在项目开发后期，我们尝试学习并使用了基于 Github 的程序开发共享和版本控制技术，显著提高了开发效率和沟通效率。