> 项目地址

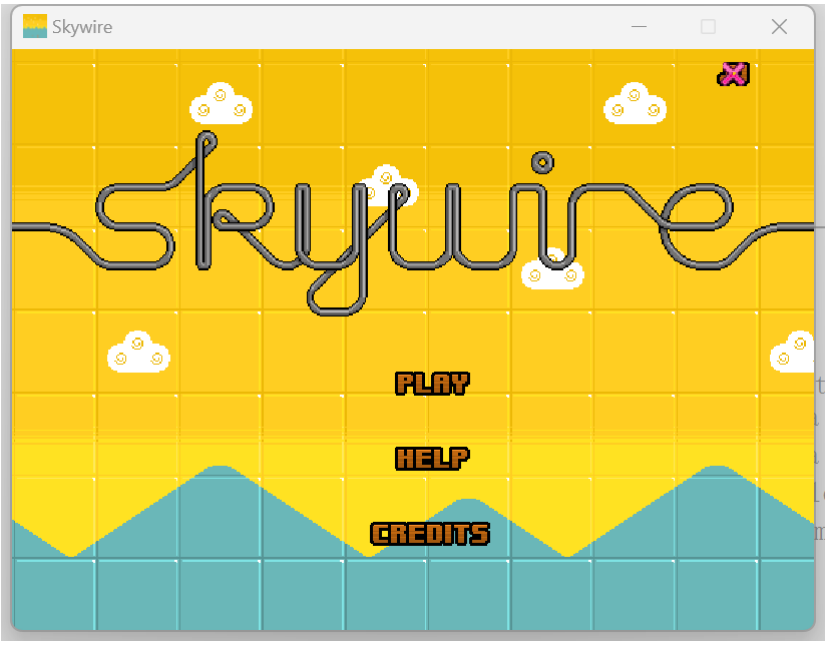
程序功能

Skywire 是一款 Nitrome 发行的动作游戏。玩家通过操作上下按钮控制一辆缆车的速度，将三名乘客中的至少一名带到终点站台。

程序核心的目的在于利用 QT 内部的函数与类以及通过解包原版 Skywire 而获得的图片、音乐等素材进行对 Skywire 游戏的**复刻和二次创造**。按照结构而言，本程序可分为两个模块，分别为**界面模块和游戏模块**，以下将分别就这两部分进行说明。

界面模块

主界面菜单

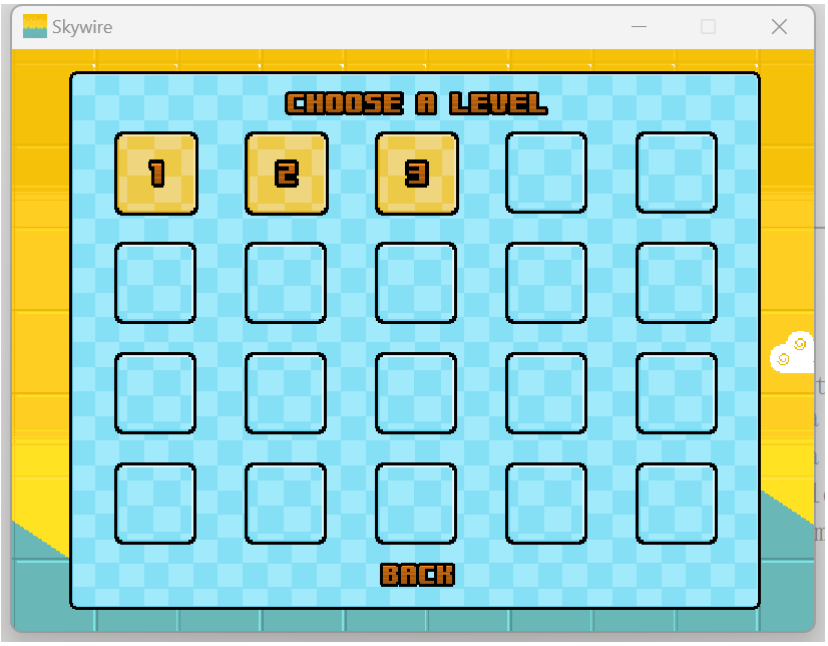


背景、字体、图标等均复刻了原版游戏。

音乐可以关闭。

有开始游戏、帮助和致谢三个选项。

点击 `PLAY` 后会选择关卡。



目前完成了三个关卡的设计（演示视频中仅包含一个关卡，录屏完成后新添加了两个关卡）。

关卡结束界面

显示存活人数、剩余时间、得分（前两者的成绩）、总得分（已玩关卡的得分之和），在顶栏左上角显示了当前的关卡 level。

有回到选择关卡界面、重玩本关、玩下一关三个选项。

游戏模块

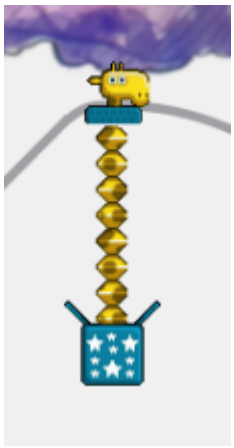
缆车与轨道

缆车运动速度由重力、摩擦力、上下按钮分别提供的正反向力决定，运动方向由轨道决定，轮子随轨道转动。

一共有三个人三条命，每次碰到陷阱减一条命，随后会有短暂的无敌时间。

陷阱

长颈鹿：不断伸缩，只有头部是有效的触碰部位。

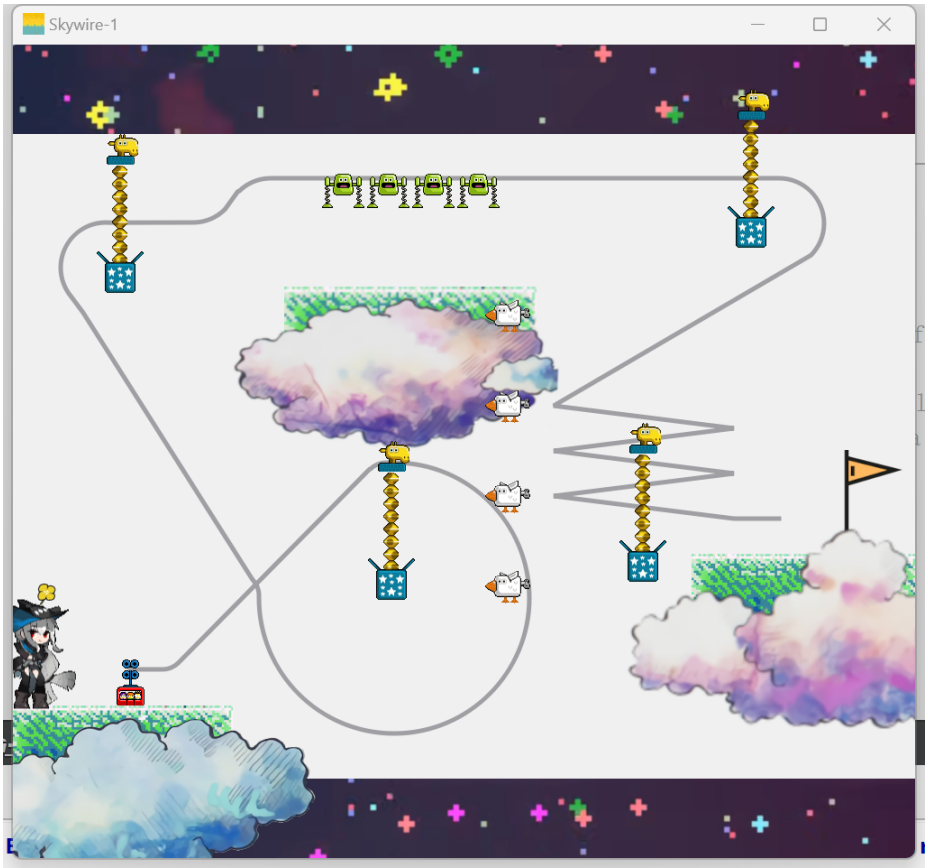


青蛙：不断跳跃。

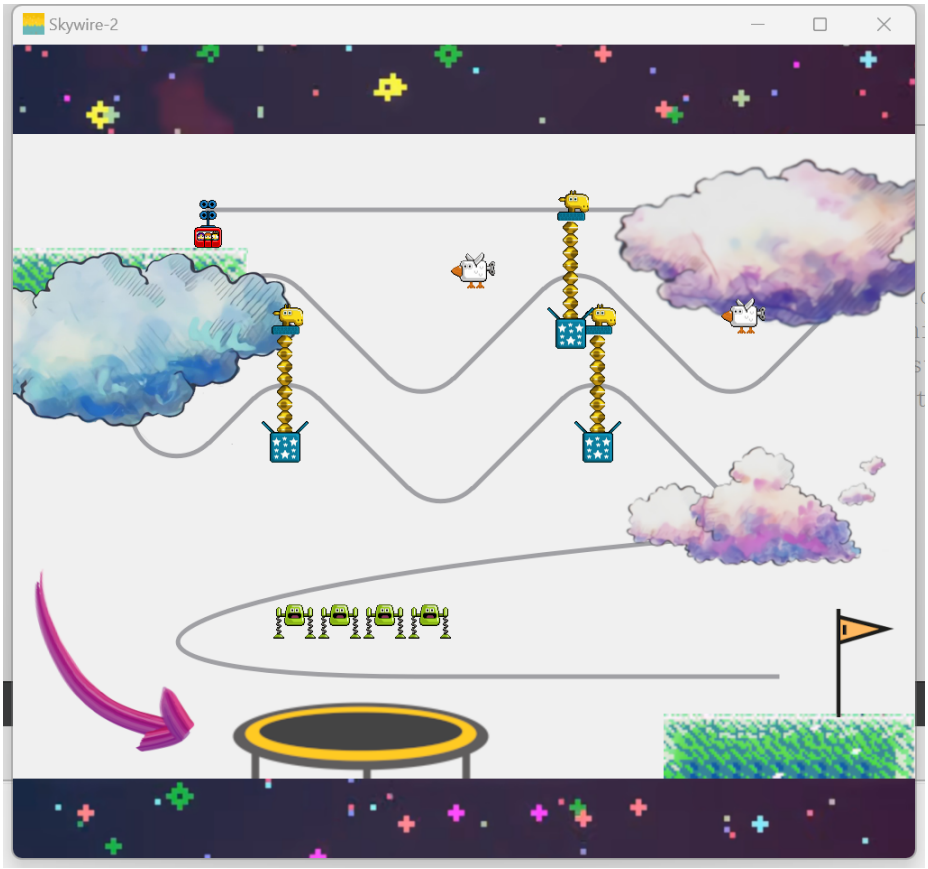
鸟：不断从右向左飞。

关卡

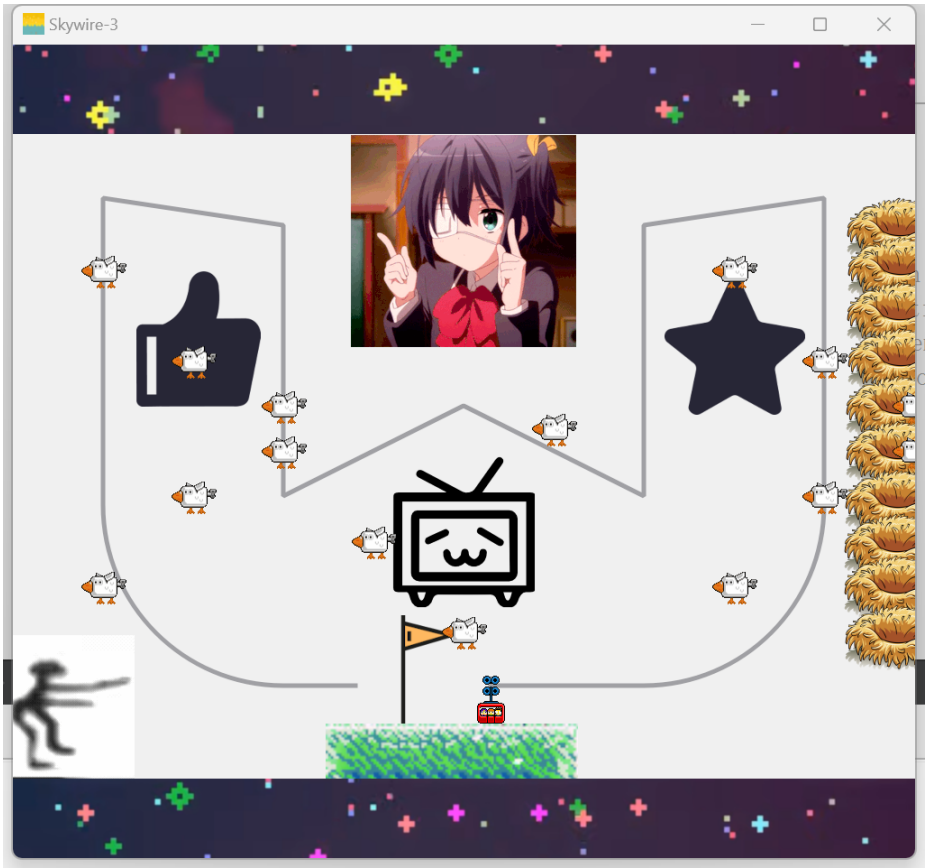
关卡 1：较为简单，用于熟悉操作。



关卡 2：难度较大。



关卡 3：通关难度较小，无伤通关难度很大。



各模块设计细节

`mainwindow.h` 和 `mainwindow.cpp`

这些文件创建了一个带有菜单、选择关卡、帮助和致谢等功能的窗口，构建了各个界面的逻辑，使用信号和槽函数实现不同界面之间的切换和交互。主要功能包括：

1. 创建了一个`MainWindow`类，继承自`QMainWindow`，作为主窗口的界面。
2. 在构造函数中，初始化界面并设置窗口的图标和标题。
3. 通过连接信号和槽函数的方式实现了菜单的功能。
4. 在选关界面中，点击不同关卡的按钮，可以进入对应的游戏关卡。
    - 进入游戏关卡后，创建一个`QStackedWidget`用于管理游戏关卡界面的切换。
    - 游戏关卡界面为一个自定义的`Widget`类的实例，根据传入的参数显示不同的关卡内容。
    - 完成游戏后，会显示得分界面`ScoreWidget`，显示游戏结果和总得分，并提供返回、重新开始和进入下一关的选项。
5. 其他界面（帮助、制作人员信息）的显示和返回功能类似，使用了类似的信号和槽函数连接方式。

`mypushbutton.h` 和 `mypushbutton.cpp`

这些文件定义了一些自定义的`QPushButton`子类，实现了不同的按钮效果和行为。主要包括以下几个类：

1. `MyPushButton`类：继承自`QPushButton`，创建一个基本的自定义按钮。在构造函数中加载正常和按下状态的按钮图像，并设置按钮的样式、图标和大小。重写了鼠标按下、释放、进入和离开事件的处理函数，实现了按钮稍微移动和图像替换的效果。

2. `ReplaceImgButton` 类：继承自 `MyPushButton` 类，创建一个可以切换图像的按钮。除了继承 `MyPushButton` 的功能外，重写了鼠标释放事件的处理函数，根据当前的状态切换按钮图像。
3. `LevelPushButton` 类：继承自 `MyPushButton` 类，创建一个可以缩放图像的按钮。除了继承 `MyPushButton` 的功能外，重写了鼠标进入和离开事件的处理函数，根据鼠标状态加载不同大小的按钮图像。

`mymenu.h` 和 `mymenu.cpp`

这些文件共同用于**主界面菜单**的类 `MyMenu` 的实现。

主要功能包括播放和暂停背景音乐，显示菜单按钮，处理按钮点击事件，并绘制背景图片。

`chooselevel.h` 和 `chooselevel.cpp`

这些文件共同用于**选择关卡界面**的类 `ChooseLevel` 的实现。

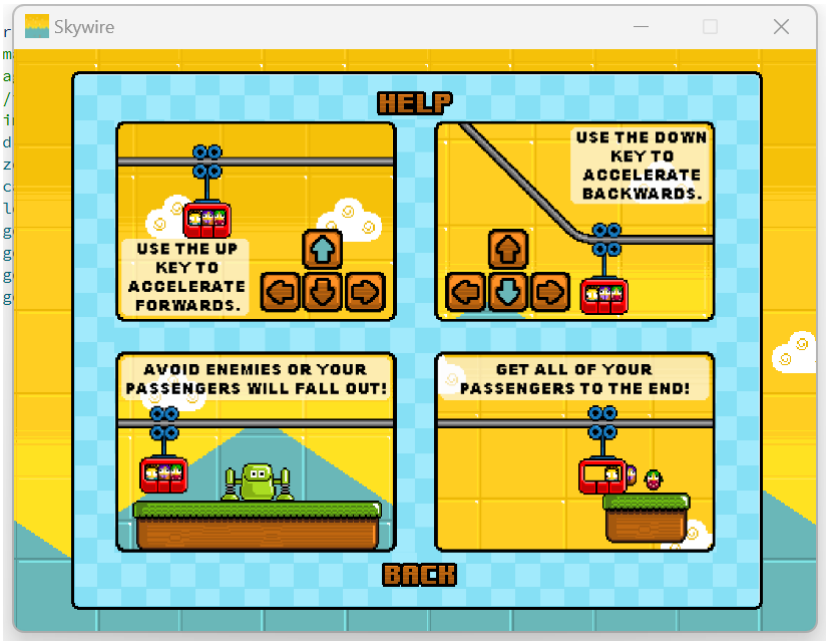
定义了一个名为 `ChooseLevel` 的类，继承自 `QWidget` 类。该类包含了构造函数、`paintEvent` 函数和一些信号等。

构造函数是一个显式构造函数，接受一个 `QWidget` 类型的指针 `parent` 作为参数，定义了选择关卡界面的布局和按钮的创建、位置设置以及信号与槽的连接。

`paintEvent` 函数是一个重写的虚函数，用于绘制选择关卡界面的背景，包括底部图片、背景图片、边框图片和标题图片。

`signals` 部分定义了一些信号，包括 `back()`、`level1()`、`level2()` 和 `level3()`。这些信号分别表示返回按钮点击、第一关按钮点击、第二关按钮点击和第三关按钮点击的事件。

`help.h` 和 `help.cpp`



这些文件共同用于帮助（HELP）界面的显示。

`credits.h` 和 `credits.cpp`

这些文件共同用于致谢（CREDITS）界面的显示。

`score.h` 和 `score.cpp`

这些文件共同用于**得分**界面的显示。

定义了一个名为 `ScoreWidget` 的自定义 `QWidget` 子类，主要功能包括：

1. 在构造函数中初始化得分相关的成员变量，创建 `back_button`、`next_button`、`again_button` 三个 `LevelPushButton` 按钮，并设置它们的位置和图像。
2. 使用 `QPainter` 在 `paintEvent` 函数中绘制背景和得分数字。背景使用一个图片进行绘制，而得分数字使用了一个包含数字图片的数组进行绘制。
3. `paintEvent` 函数根据游戏的得分和时间计算得分数字，然后将得分数字绘制在界面的适当位置。
4. 在构造函数中创建了一个背景音乐播放器，并设置了背景音乐的播放模式和音量。当 `back_button` 按钮被点击时，停止播放背景音乐，并通过发射信号告知其他部分返回上一个界面。

`next.h` 和 `next.cpp`

这些文件仅用于第三关后暂无下一关卡的显示界面。

`add_map.h`

该文件用于**轨道和陷阱**的设置。

轨道采用 `QPainterPath` 类，通过 `moveTo`、`lineTo`、`arcTo`、`cubicTo` 四个函数进行延伸。

陷阱采用 `QVector` 类。

`trap.h`

该文件用于表示游戏中的**陷阱**。

定义了一个名为 `Trap` 的类。它表示一个陷阱，具有以下成员变量：

- `x` 和 `y`：表示陷阱的坐标位置。
- `name`：表示陷阱的类型名称。
- `label`：一个 `QLabel` 对象，用于在界面上显示陷阱的图像。
- `movie`：一个 `QMovie` 对象，用于播放陷阱的动画效果。
- `radius`：表示陷阱的半径大小。
- `angle`：表示陷阱的角度。

在构造函数中，可以传入陷阱的初始位置（x 和 y 坐标）、类型名称和角度。根据类型名称，设置相应的半径大小。

`updatePosition` 函数用于更新陷阱的位置，接受新的 x 和 y 坐标作为参数，并更新陷阱对象的位置以及图像显示的位置。

在析构函数中，释放了陷阱对象所占用的内存，删除了 `label` 和 `movie` 对象。

`widget.h` 和 `widget.cpp`

这些文件共同提供了处理**用户交互、游戏逻辑和绘制界面**的功能。

定义了一个名为 `Widget` 的类，继承自 `QWidget`。它是游戏界面的主要窗口，用于显示游戏内容和处理用户交互。

主要成员变量和函数包括：

- `_people` 和 `_time`：表示游戏中的剩余人数和剩余时间。
- `invince_button`：一个 `ReplaceImgButton` 对象，用于触发无敌状态。
- `invincible()`：处理无敌状态。
- `paintEvent()`：绘制缆车和轨道，并检查与陷阱的碰撞。
- `paintPath()`、`paintCar()`、`paintTrap()`：绘制游戏中的缆车、轨道和陷阱。
- `crushTrap()`：处理缆车与陷阱的碰撞。
- `keyPressEvent()`：处理用户按键操作，根据按下的键更新缆车的速度。
- `updatePosition()`：根据缆车的速度更新缆车沿路径的位置。
- `updateSpeed()`：根据重力和摩擦力更新缆车的速度。
- `loseLife()`：当缆车与陷阱碰撞时调用，减少剩余的生命数量，启动无敌状态，播放声音并显示死亡动画。
- `Youlose()`、`Youwin()`：处理游戏失败和胜利的情况。
- `reset()`：在游戏结束后调用，无论输赢，重置游戏状态并为新一轮做准备。
- `toscore()`：切换到得分界面。
- 其他成员变量和函数用于处理游戏逻辑，包括计时器、道具效果、背景贴图、音效等。

- 构造函数：执行各种初始化和设置操作，例如设置小部件的大小、创建计时器、设置媒体播放器、显示标签和图像以及连接信号和插槽。

小组成员分工

组长：徐皓天

主要负责设计项目功能，解包原版 Skywire，撰写作业报告等工作。

组员 1：杨世航

主要负责设计游戏逻辑，编写游戏模块代码，设计关卡等工作。

组员 2：刘炫辰

主要负责设计菜单布局，编写界面模块代码，组装代码等工作。

总结与反思

在本次 QT 大作业的完成过程中，我们成功复刻了 Skywire 的各项基础功能，并在此基础上对游戏逻辑、关卡设计等部分进行了二次创造，将许多原创的 idea 进行了实现。

在实现的过程中，我们遇到了许多挑战和困难。其中一些成功解决，包括原版游戏的素材获取、轨道的刻画、陷阱的动效等。还有一些没有解决，比如原版「保持缆车在界面中心，镜头随缆车移动」的效果。

通过这次实验，我们学到了很多有关 QT 的知识，增强了阅读技术文档的能力，积累了宝贵的经验，同时大大锻炼了编写大型项目代码的能力，对面向对象的理念也有了更深的理解。另一方面，我们学会了如何进行分工协作，以达到效率的最大化。