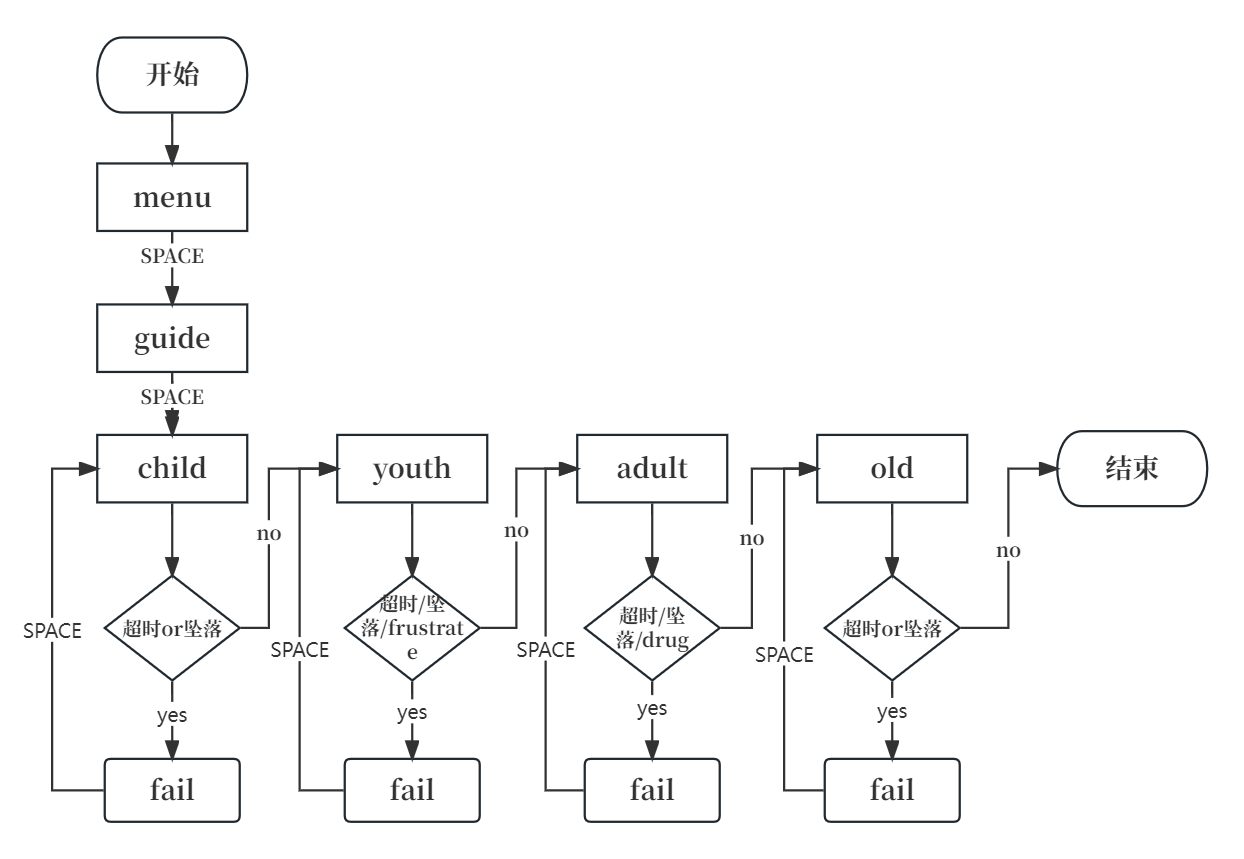
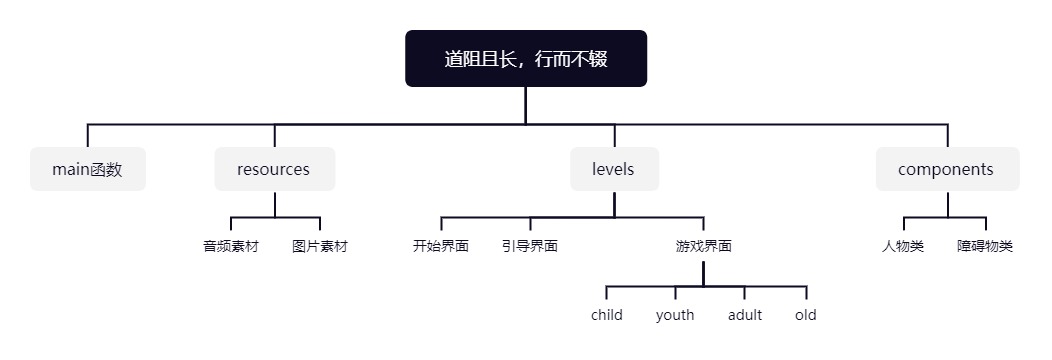
**1．总体设计**

**1.1 业务流程**

上图为游戏逻辑流程图，游戏大体为顺序结构，各界面都可以按下ESC键退出游戏。经过开始界面和引导界面后，游戏开始。前一关游戏在规定时间内通关则自动切换至下一界面。若玩家在某阶段失败，则会跳出提示界面，指示玩家按下空格键重新开始本阶段。经过全部四个阶段游戏通关，不能中途开始或回溯。

**1.2 总体架构**

游戏模块组成结构图展示如下：



**2．程序描述**

**2.1功能**

程序可以实现的功能有：建立对象并在主函数中应用；利用pygame精灵类实现人物和障碍物之间的碰撞检测；通过检测键盘操作实现人物的左右移动和跳跃；

**2.2性能**

性能良好，受制于某些阶段素材较多以及pygame的相关特性，图片渲染等过程会咀嚼CPU，导致游戏产生掉帧现象，有待后续通过停止无用的渲染更新、及时清理无用的精灵、利用convert()函数导入图片等途径优化性能。

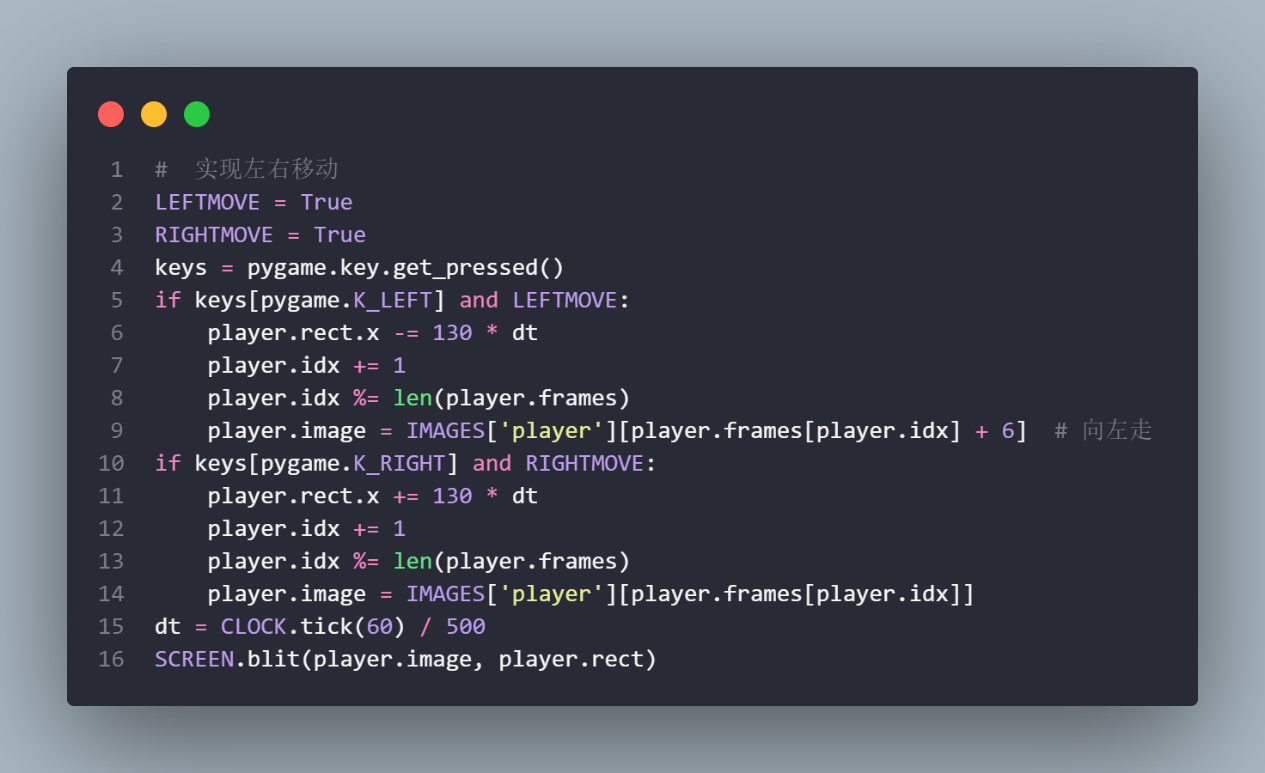
**2.3输入项目**

本游戏无需输入内容即可直接游玩。

**2.4输出项目**

通过键盘操作，在程序中可实现图形界面的逐帧输出，表现为游戏画面的动态变化；经过通关、死亡判定会显示相关静态界面；在源文件中打印相关变量可在终端实现人物实时坐标、移动速度等的显示。

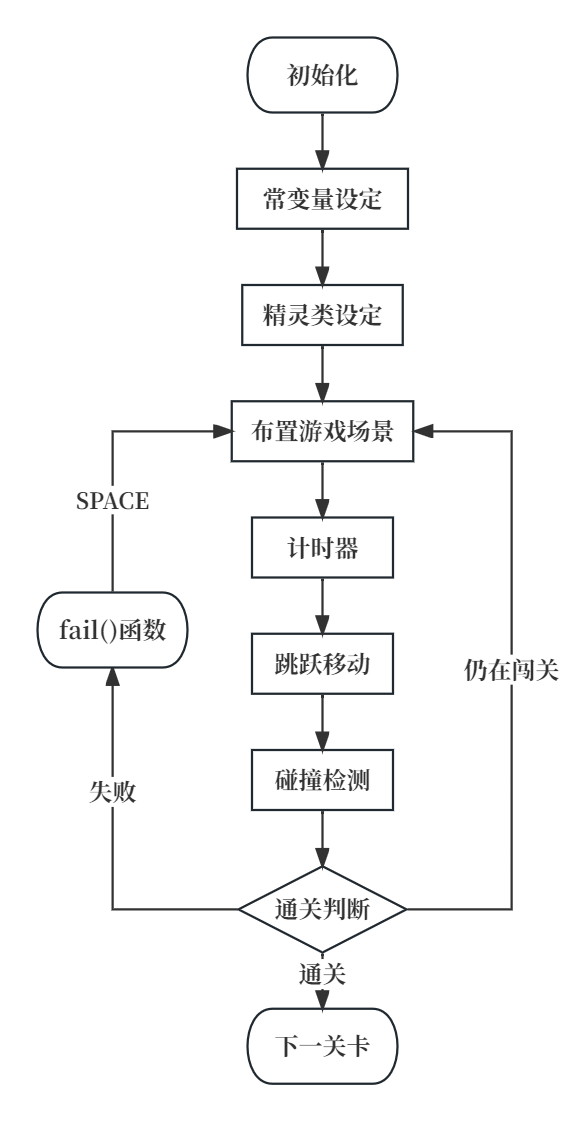
**2.5算法**

 **人物移动：设定bool变量以判断移动状态，根据移动状态执行操作。此外，移动时还需要展现人物动作变化，即图片的切换。我在player类中设定了idx变量和frames列表，随着时间推进，idx自增并与len(frames)取模，实现人物动作的动态变化。**

**碰撞检测：**利用pygame精灵类的pygame.sprite.spritecollideany()接口，不足之处是无法判断两精灵是从何方向碰撞的。解决方法：对横纵向的各种情况分别判断，横向碰撞则横向速度置零，从下向上碰撞则纵向速度置零开始下落，从上向下碰撞则速度置零静止不动。

玩家掉落的碰撞判断：使玩家向下落一像素，若未与平台发生碰撞且未处于跳跃状态且未处于地面上，则进行自由落体运动，随后玩家位置回升一像素。这样可以实现真实物理环境下玩家的跳跃和避障。

人物跳跃：

**2.6程序逻辑**

**2.7接口**

大多调用pygame中的接口。

显示界面及声音播放，如pygame.image.load(), pygame.time.set\_timer(), pygame.font.SysFont(), pygame.display.update(), pygame.mixer.Sound()等。

精灵类，如pygame.sprite.Group(), pygame.sprite.spritecollideany(), group.add(), group.draw()等。

输入状态获取，如pygame.key.get\_pressed(), pygame.event.get(), pygame.quit()等。

**2.8存储分配**

游戏对于存储要求并不高，故采用静态存储分配的方式。

使用全局变量存储经常使用的常量，以优化代码可读性，如需要使用的图片与音频名称、各种颜色的RGB值、游戏帧率、画面长宽等。

在各函数中设置局部变量以实现游戏结束判断、人物移动判断、关卡内各种buff的获取情况等，具体关卡具体设置。

**2.9限制条件**

系统需为Windows系统，安装pygame模块以对代码进行调试。受制于时间限制，游戏中的功能、buff等有待丰富和优化，游戏UI设计还可以进一步完善，素材风格有待进一步统一。

**2.10测试要点**

使用空格键进行游戏界面切换，ESC退出游戏，左右方向键控制人物左右移动，上键控制人物跳跃。

测试时向下跳出地图则会触发死亡提示，左右边界有空气墙。若玩家从两侧碰到平台，则会水平速度置零开始自由落体直至碰到下一平台或死亡。

**3. 系统数据结构设计**

**3.1逻辑结构设计**

|  |  |  |  |
| --- | --- | --- | --- |
| 标识符 | 数据结构名称 | 定义 | 备注 |
| IMAGES[name] | 字典 | 每一个图片名对应该图片的相对地址 | 不定长，随添随用，图片名为存储地址中去掉后缀的名称 |
| frames[] | list数组 | 人物移动时图片周期性变化的频率 | 每个人物类均有各自的图片变化的频率，通过idx变量形成索引，访问 |
| keys[] | 数组 | 获取当前按下的键盘按键 | 随时变化 |
| brick\_group[] | 精灵组 | 存储游戏画面中的障碍平台 | 变量包含平台当前的横纵坐标和长宽等数据 |
| player\_x\_range[] | 数组 | 人物移动x坐标范围 | 定长，在guide\_window中使用 |

**3.2物理结构设计要点**

本系统内各数据结构均较简单，均为顺序存储，访问为直接访问，IMAGES字典为全局变量，其余均为各函数和各类内定义的局部变量；人物类中的frames数组存储在component\player文件中，在各人物类中均有定义。其余各数据结构均存储于main函数中。设计简单的数据结构的考虑，一是保持代码的简洁程度以便后续维护，二是该程序对内存要求并不高，顺序存储足以完成游戏中的相关操作。

frames列表通过人物类中定义的idx变量实现索引功能，每次循环中，idx自增并与一次循环的总帧数取模，得到当前应当显示的图片，这样简化了代码，避免冗长的代码段重复。