

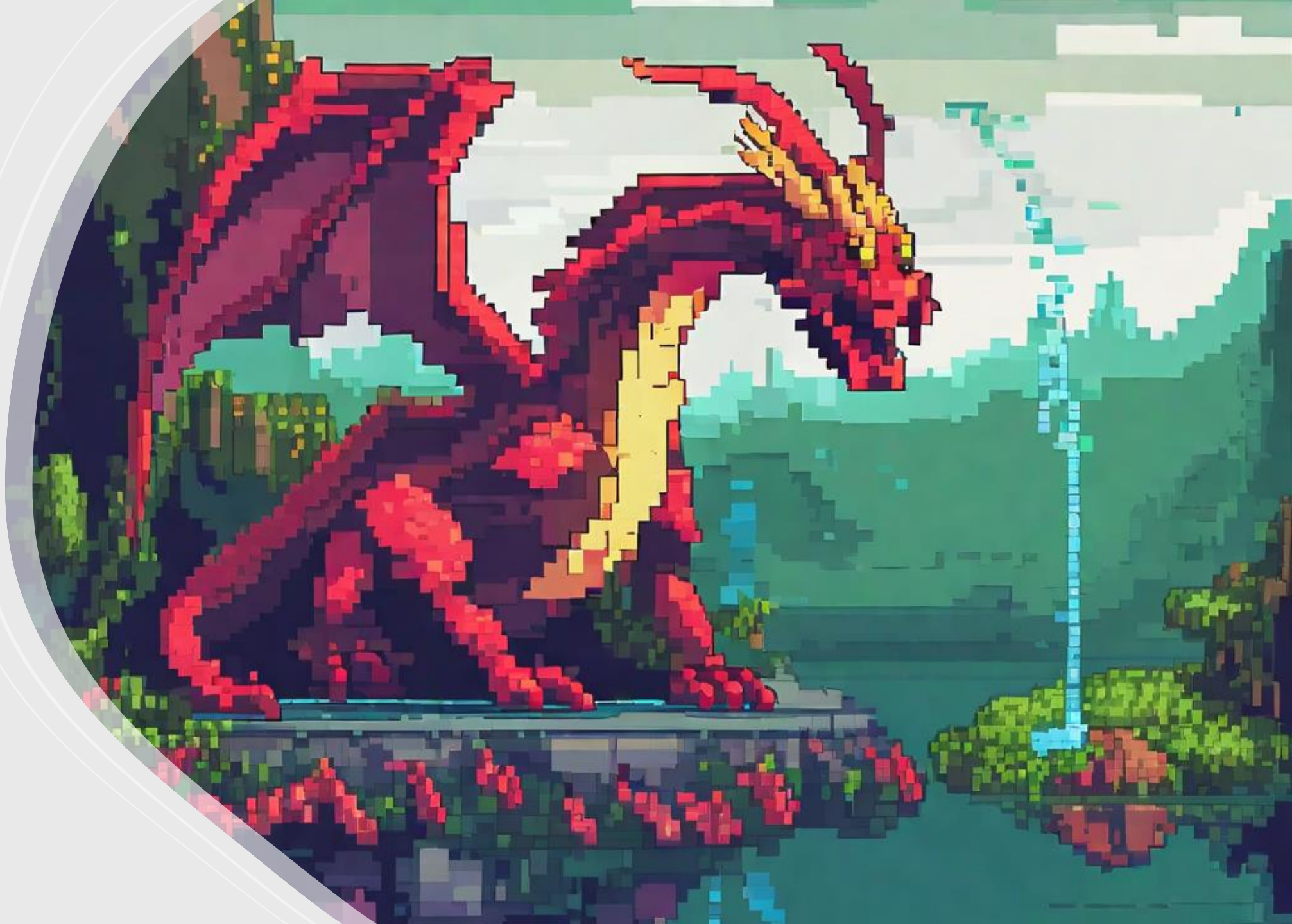
A pixel art landscape featuring a bright blue sky with white clouds, green rolling hills, and several trees with green foliage and brown trunks. The style is reminiscent of early computer graphics or indie game art.

北理润

BITRun

部分绘图: Stable Diffusion XL

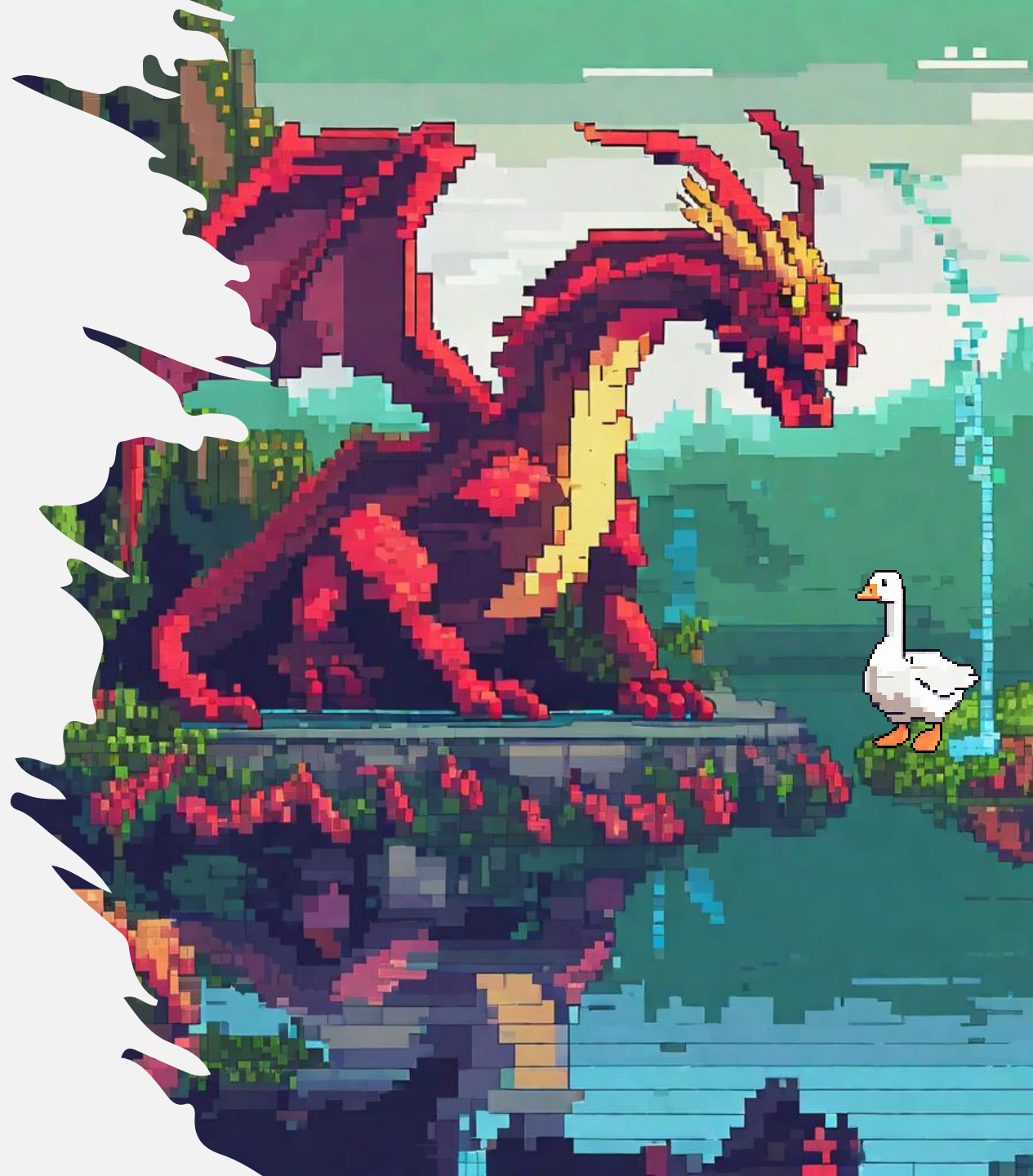
游戏 背景



很久很久以前,在一个叫百丽宫的地方生活着一群无忧无虑的大鹅,然而东食堂的出现打破了往日的宁静,他们为了得到鹅腿不择手段。

鹅们召唤出了北湖中的神兽——百丽宫的恶龙,恶龙答应帮助大鹅,但要一个东西作为交换——张博士的汇编期末试卷。

为了鹅族的存亡,你——鹅长老,决定润出百丽宫,这注定是一场艰难的长征。。。。



北理润

启动！



然而

鹅是没有手的

按不了空格



你说得对
但是
「北理润」
是……

但鹅会叫!

北理润

再次启动！



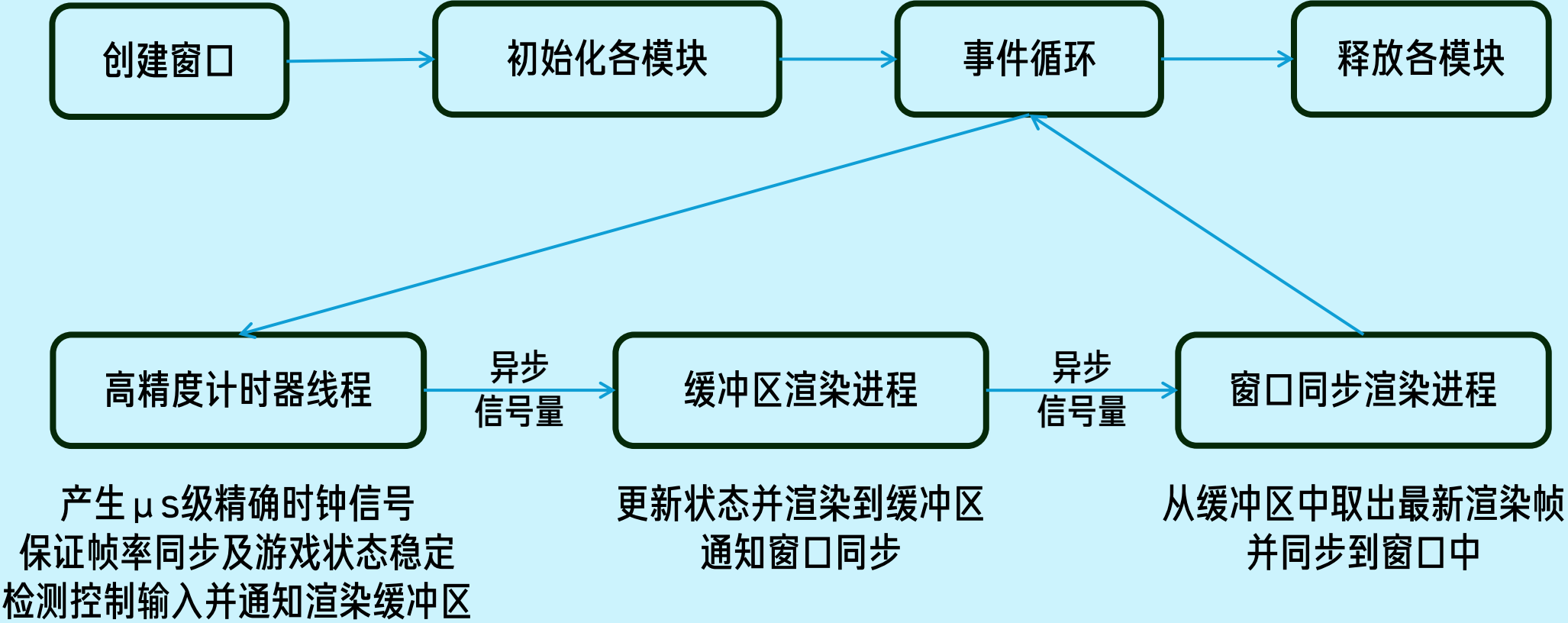
弱小和无知
不是生存的障碍
傲慢才是

于是我们可以骄傲地宣布
我们（可能是）首个不需要手就能玩的游戏



主流程控制 &渲染模块

主流程控制



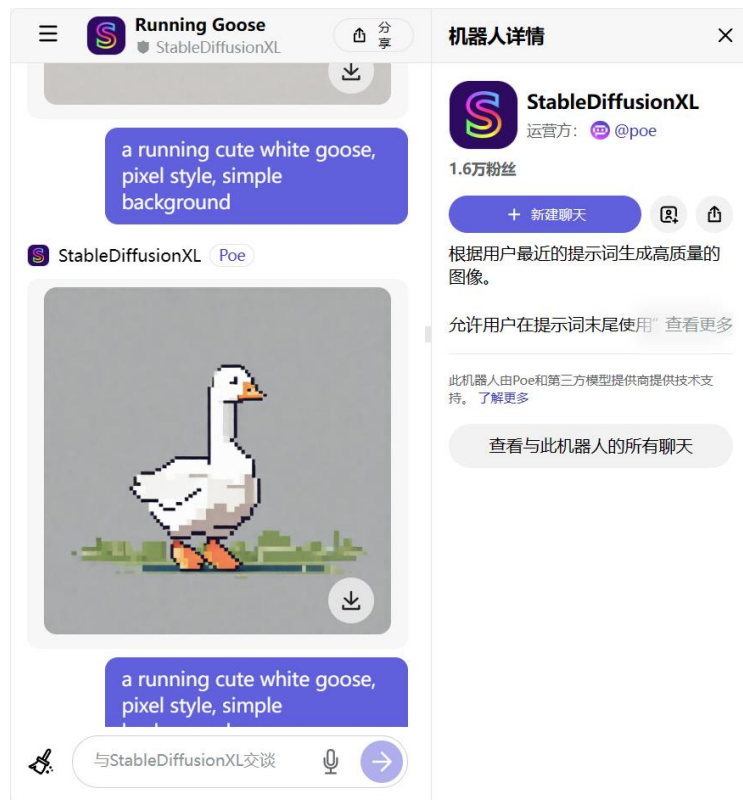
渲染流程

1. 从队列中依次取出RenderObject渲染对象
2. 索引渲染对象对应图像
3. 按照优先级变换坐标并渲染到缓冲区画布上
4. 渲染其他元素如分数、游戏提示等

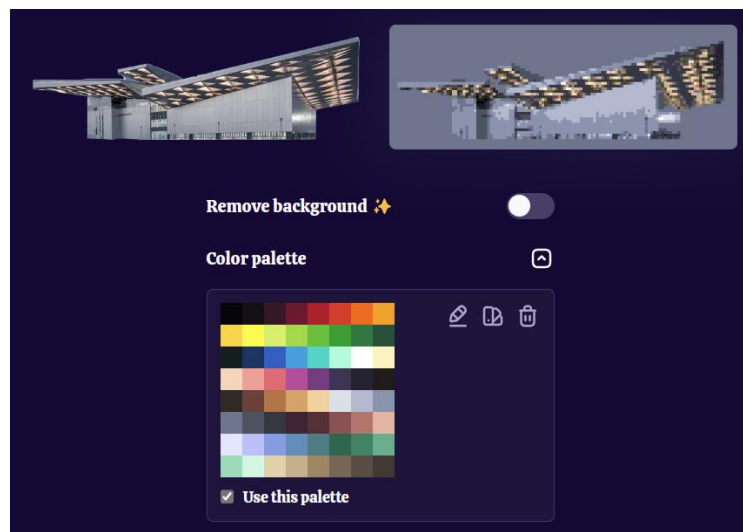
```
; 图片结构体
Image STRUCT
    id DWORD ? ; 资源ID
    mask_color DWORD ? ; 透明色
    w DWORD ? ; 宽度
    h DWORD ? ; 高度
    h_dc DWORD ? ; DC句柄
    h_bmp DWORD ? ; BMP句柄
    a_mask DWORD ? ; w*h的数组 每个元素为BYTE 表明是否有像素
Image ENDS

; 绘制元素结构体
RenderObject STRUCT
    obj_id DWORD ? ; 元素的ID
    x DWORD ? ; 画布上的X坐标
    y DWORD ? ; 画布上的Y坐标
    z DWORD ? ; 优先级,取值为0,1,2, 优先级越高越先绘制
    p_image DWORD ? ; 图片指针
    lasttsp DWORD ? ; 元素最近更新的时间 单位毫秒
    phsx REAL4 ? ; 元素物理X坐标
    phsy REAL4 ? ; 元素物理Y坐标
    vx REAL4 ? ; 元素X方向速度
    vy REAL4 ? ; 元素Y方向速度
RenderObject ENDS
```

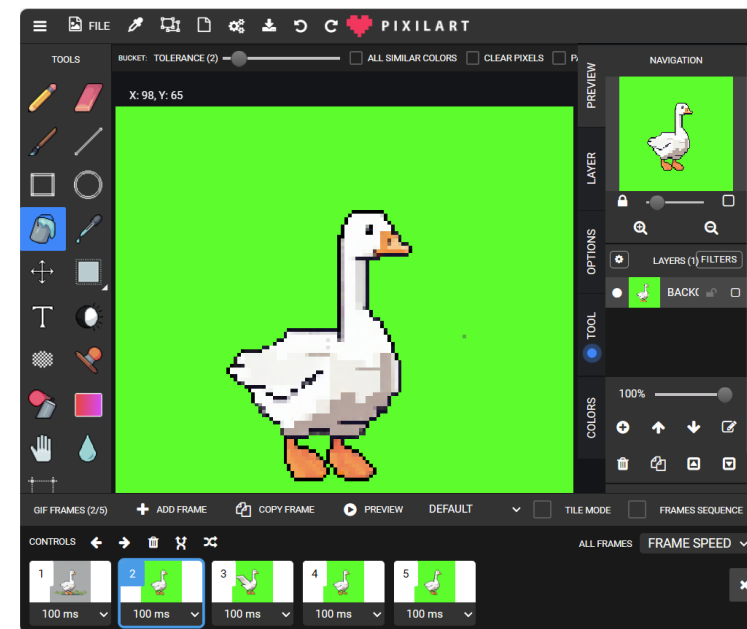
图像资源设计三板斧



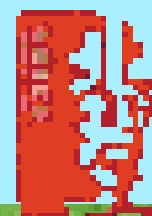
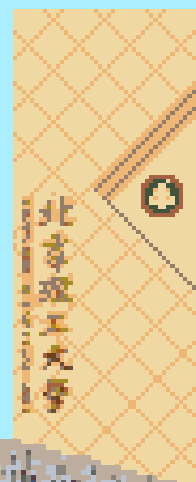
AI绘图



像素配色转换



人工调整绘制



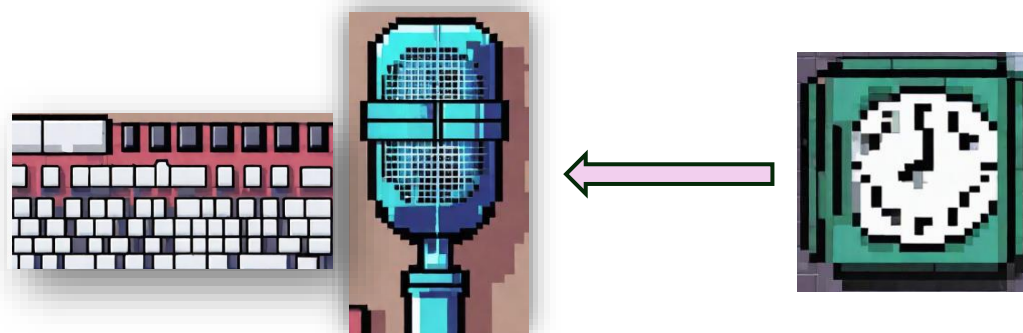


状态处理 &更新设计

State handling of the game

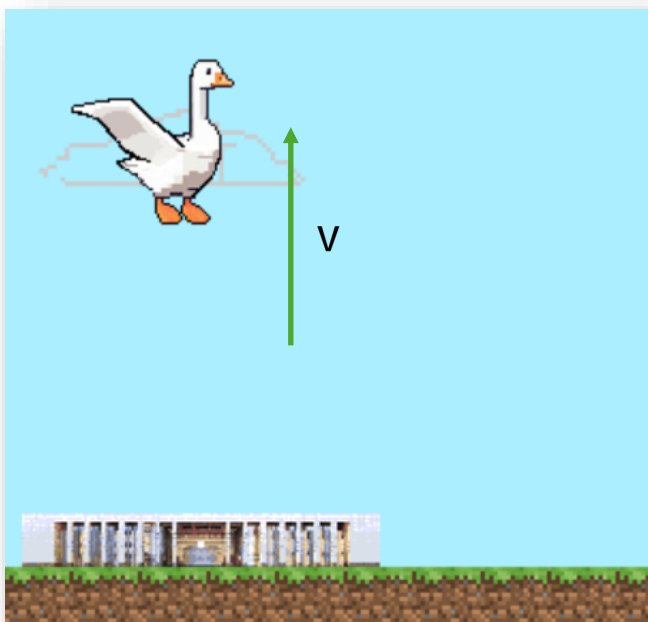
状态更新主要函数

在时钟线程中触发`_check_key_down()`函数，检查外界输入，并更新游戏内部状态



在绘制线程中则触发`_state_update()`函数，通过计算出所有游戏对象：

障碍物，背景，鹅，云朵等的下一时刻的位置
对其进行更新



_state_update() 函数

```
; 更新状态
_state_update PROC uses ebx esi
    local @collide
```

```
...
```

```
; 这里更新所有的对象运动状态，如果发生了碰撞则终止
mov esi,$state.p_a_render_object
mov ecx,0
.while ecx!=$state.render_object_size
    invoke _update_render_object,esi
    ...
.endw
```

```
.if @collide
    ... 更新游戏结束状态
.endif
```

```
; 增加全局速度，同时加速所有物体
invoke _accelerate_all_obj
```

```
; 根据是否发生碰撞更新绘制列表
invoke _update_render_list,@collide
```

```
; 根据绘制状态添加屏幕外准备出现的障碍物
invoke _update_obstacles
```

```
ret
_state_update ENDP
```

依次遍历绘制集合，通过调用
_update_render_object函数，
对每个游戏对象依照其速度与位置，
进行下一个状态的更新。当更新的时候检测与其他对象的碰撞

_update_render_list用于剔除所有
无需再绘制集合之外的元素

_update_obstacles则根据当前的绘
制集合以及游戏状态，添加未来需要出
现的游戏对象

_state_update() 函数

与绘制的主要数据结构绘制列表\$state.p_a_renderlist



绘制列表中的元素为RenderObject 结构体。每个结构体中维护了每个待绘制图像的绘制坐标以及需要使用的图片，和最为重要的运动信息。通过绘制列表我们就可以方便的对全局绘制进行操控。

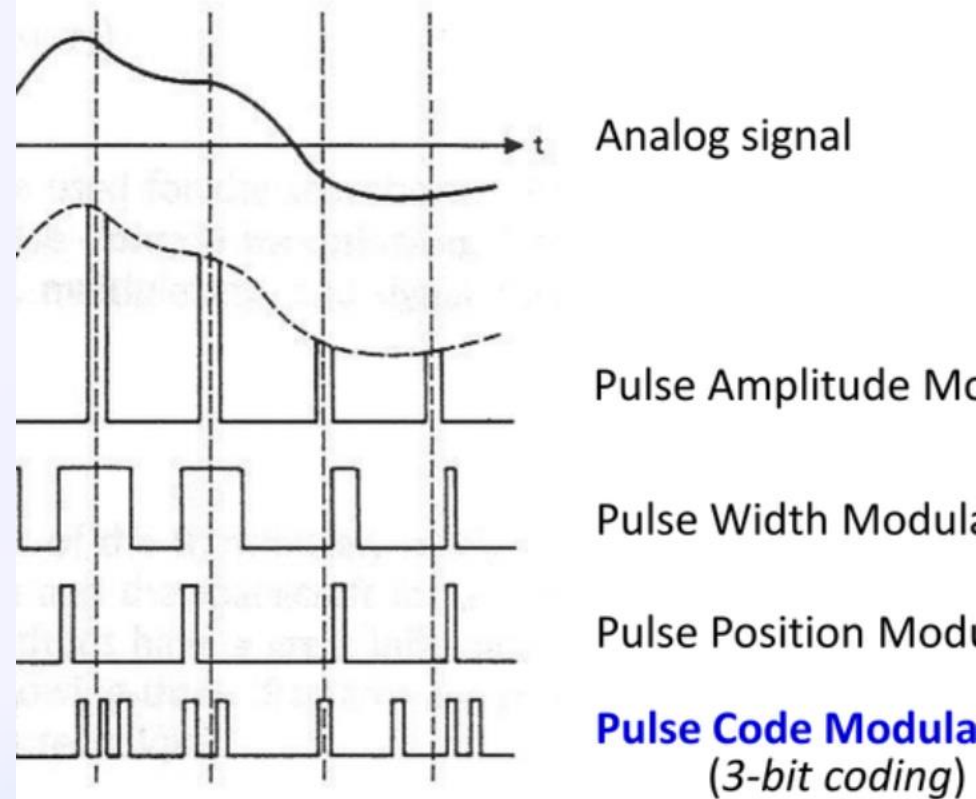
```
; 绘制元素结构体
RenderObject STRUCT
    obj_id DWORD ? ; 元素的ID
    x DWORD ? ; 画布上的X坐标
    y DWORD ? ; 画布上的Y坐标
    z DWORD ? ; 优先级,取值为0,1,2, 优先级越高越先绘制
    p_image DWORD ? ; 图片指针
    lasttsp DWORD ? ; 元素最近更新的时间 单位毫秒
    phsx REAL4 ? ; 元素物理X坐标
    phsy REAL4 ? ; 元素物理Y坐标
    vx REAL4 ? ; 元素X方向速度
    vy REAL4 ? ; 元素Y方向速度
RenderObject ENDS
```

脉冲编码调制（PCM）原理 和Winmm中wave系列函数 调用

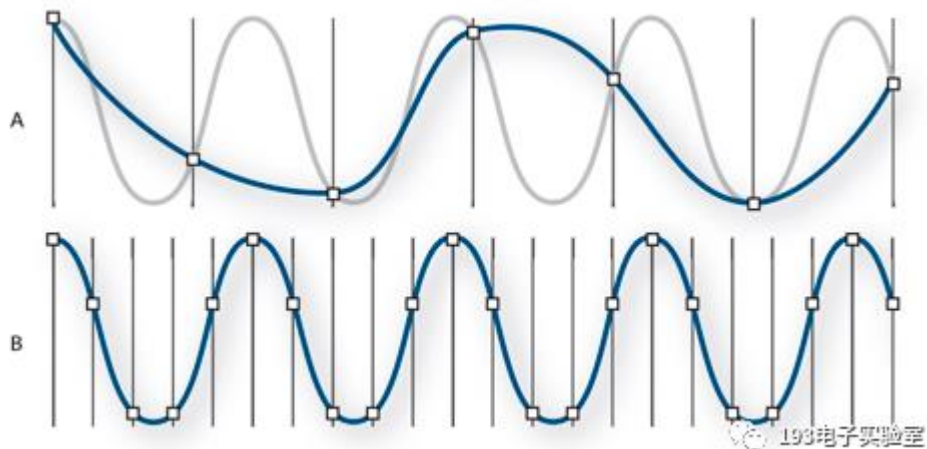
脉冲编码调制（PCM）是一种数字信号传输技术，将模拟信号转换为数字形式。Winmm中的wave系列函数是在Windows操作系统中用于处理音频数据的API。

Pulse Code Modulation

ES442 – Spring 2016



脉冲编码调制 (PCM)



• 越高，越真实



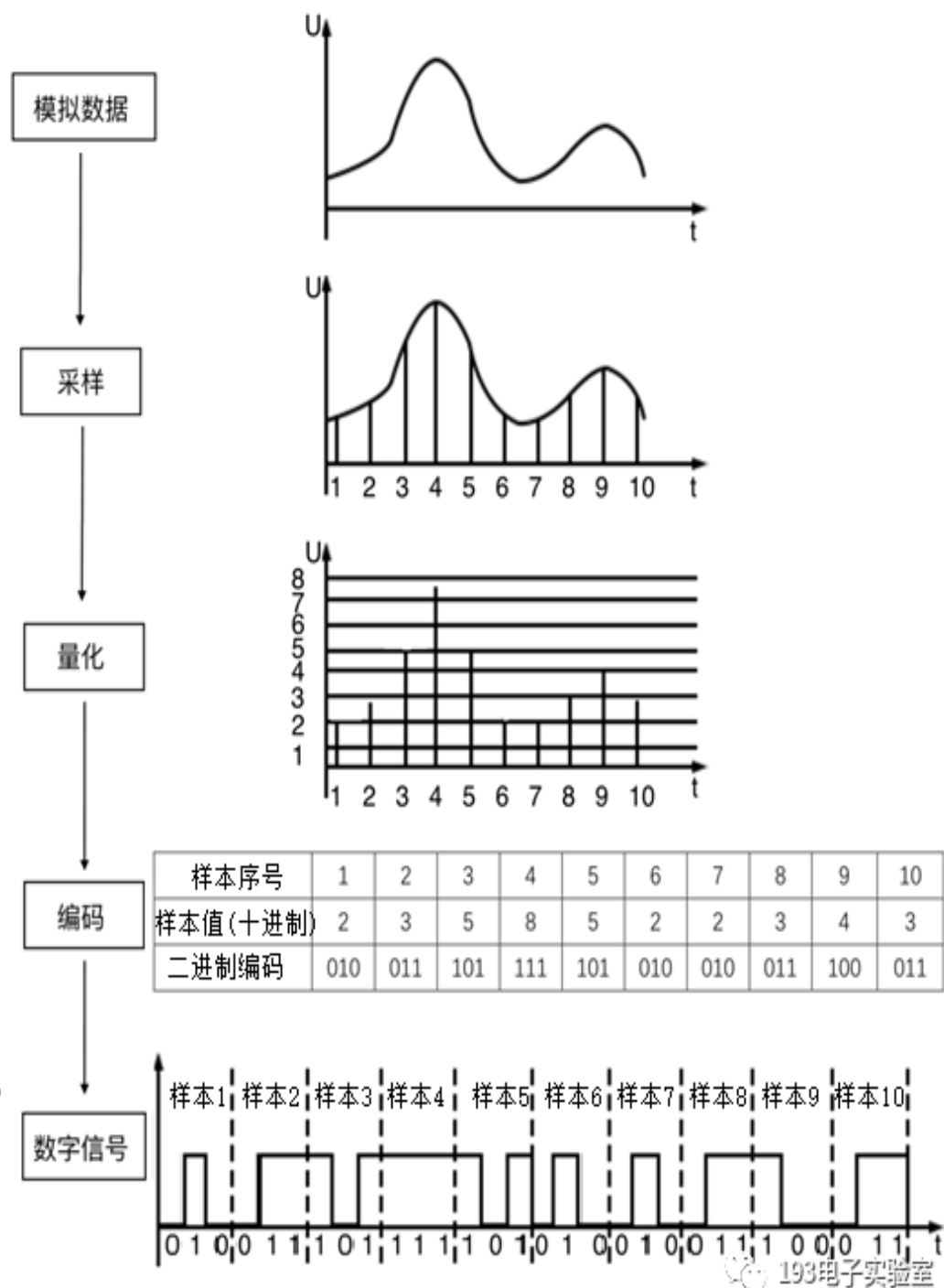
8-bit

binary	0010 0000	1010 0000	...
decimal	32	-96	...

• 越高，越精细



$$N_{dB} = 10 \log_{10} \left(\frac{p_i}{p_o} \right)$$



Winmm中wave系列函数概述

waveInOpen

打开音频输入设备，并指定音频输入的格式和参数。

waveInStart

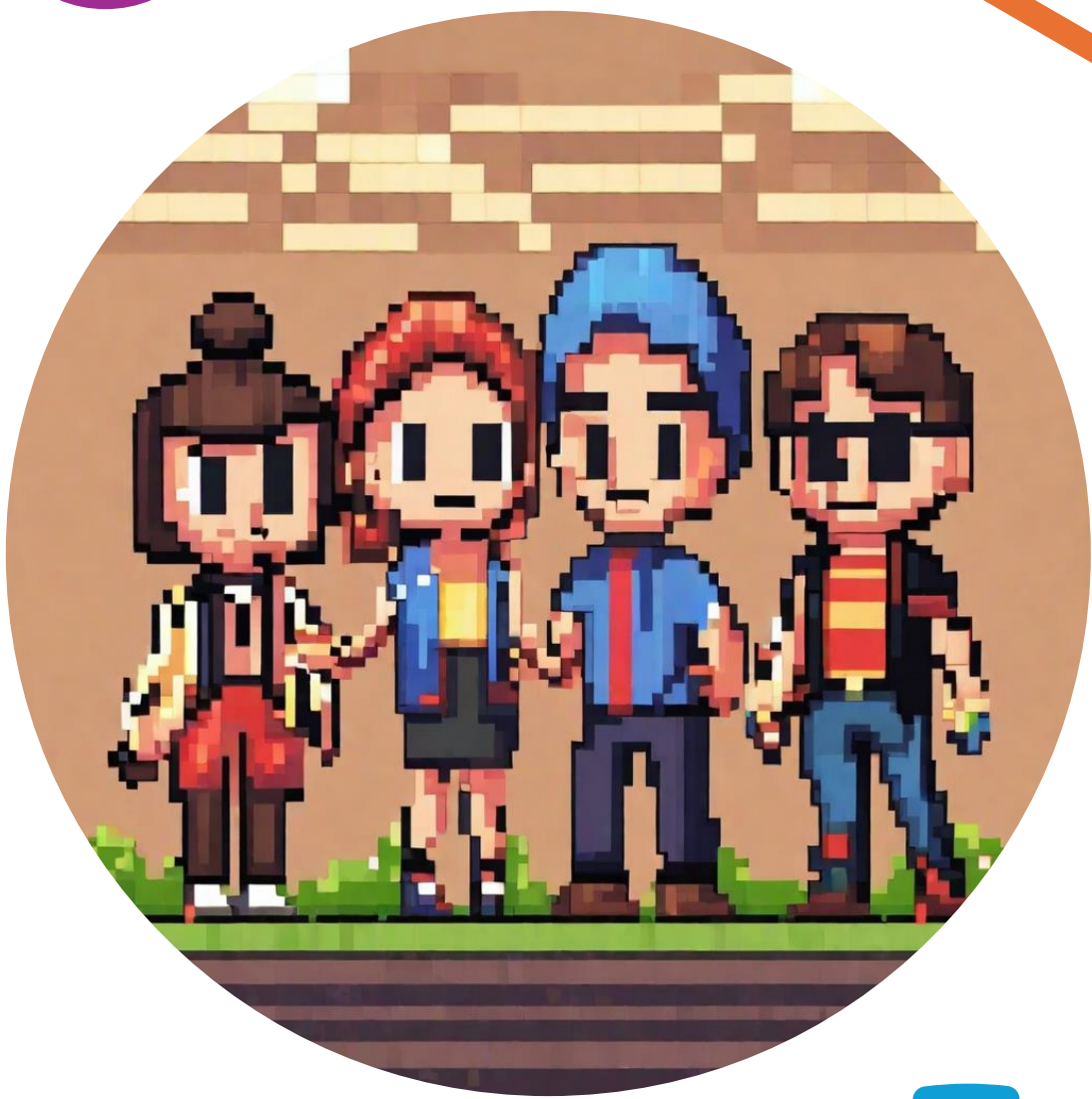
开始从音频输入设备中读取音频数据。

waveInStop

停止从音频输入设备中读取音频数据。

waveInAddBuffer

将缓冲区添加到音频输入设备的输入队列中。



小组合作

通过事先制定编码规范、进行模块化设计，我们通过Git实现了高效的版本控制及团队合作。

项目亮点

良好的人机交互设计 简洁好玩

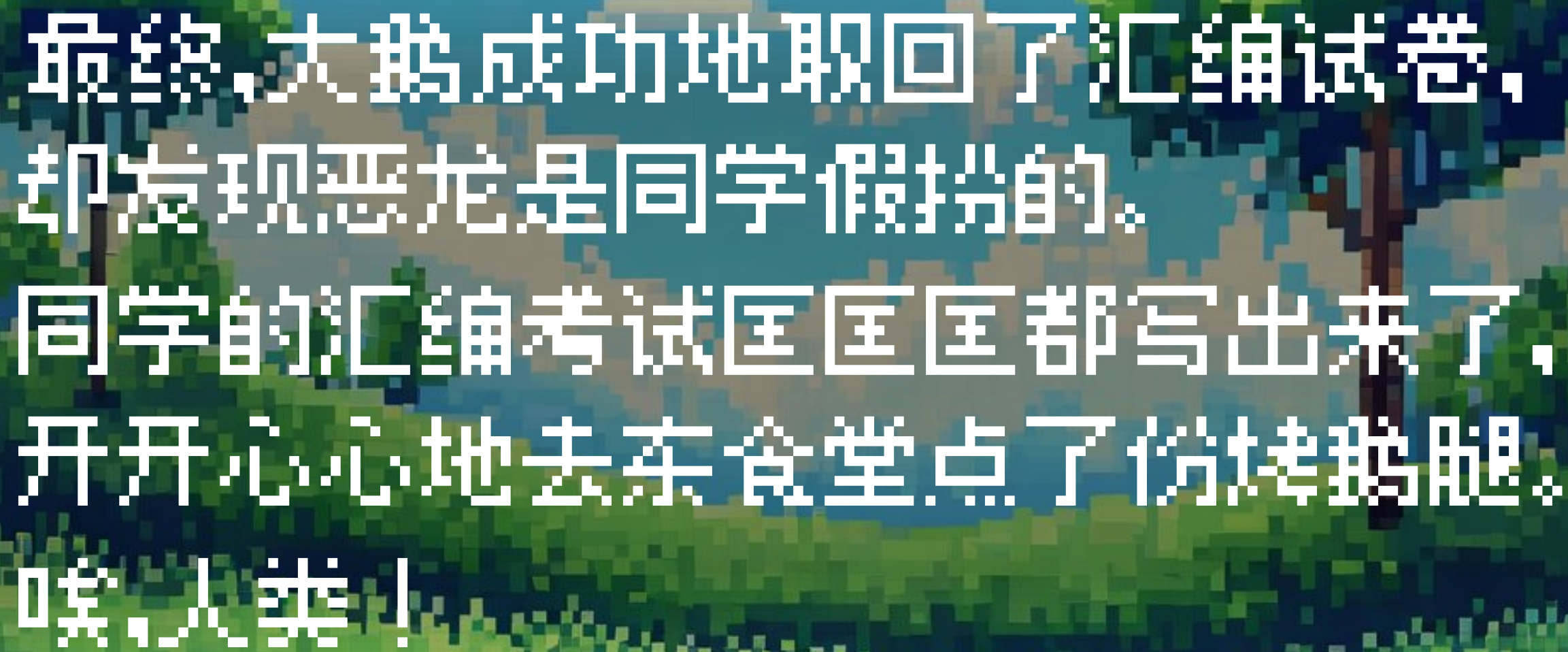
具有创造性的美术设计

清晰的代码和运行流程

高效的团队协作

小而美（单exe，低空间、CPU、内存占用）



The background is a pixel art illustration of a landscape. It features a dirt path that winds through a green field. On either side of the path are several trees with green foliage and brown trunks. The sky is a light blue with some white, pixelated clouds. The overall style is reminiscent of early computer graphics or video game environments.

最终,大鹅成功地取回了汇编试卷,
却发现恶龙是同学假扮的。
同学的汇编考试匡匡匡都写出来了,
开开心心地去东食堂点了份烤鸭腿。
唉,人类!