# Py(π)酷跑

Presented by 无魏三国

**刘（备）**子欣　**孙（权）**美琪　**（周）俞**梦婕

# CONTENT 目录

PART 01

# 设计背景

BACKGROUND

**灵感来源： Chrome 404 小恐龙**

# 设计背景

## Chrome 404 小恐龙
## Chrome Dino

**当用户尝试在与互联网断开连接时访问网站时会出现这种游戏。**Chrome 当中的恐龙游戏是一个简单的无限跑步游戏，它会让用户跳过仙人掌，并闪避障碍物，游戏为用户提供基本控制，按空格键跳转（并开始游戏）向上箭头跳跃。目标是在互联网重新开始工作之前让用户打发时间。

**chrome://dino**

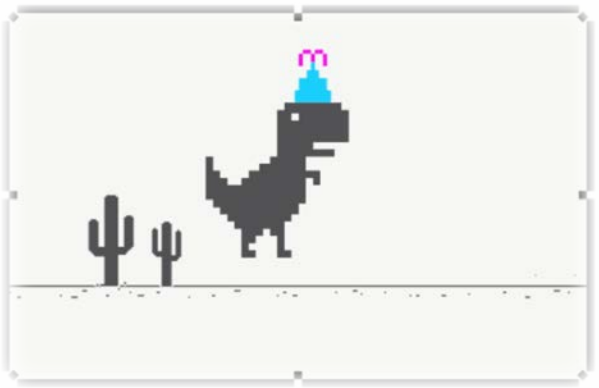
# 设计背景

## 发布背景

2014年“你掉线了”页面复活节彩蛋

2014.9 第一次提交

2014.11 扩展到了全部平台



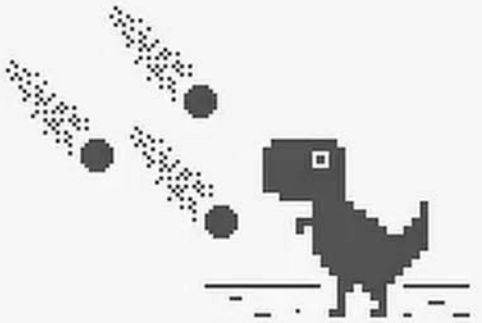

## 无限跑酷的终点

设置的最大值是 1700 万年，这个时间是霸王龙在地球上生存的时间。

“但是我觉得你的空格可能玩不了那么久。”

## 受欢迎度

目前每月包括电脑和移动端一起总共有 2.7 亿次访问。



PART 02

# 设计概念

CONCEPT

**关键词： 涂鸦 简约 无限π 跑酷**

# 设计概念

CONCEPT

## 涂鸦

模拟牛皮纸上所涂鸦的角色形态与背景
（类似于Doodle jump）

## 无限

利用π的无限性来设置障碍

## 简约

画风与操作简约，便于玩家上手

## 难度

游戏难度会随着游戏时间
&分数的增长而增加

PART 03

# 项目准备

PREPARATION

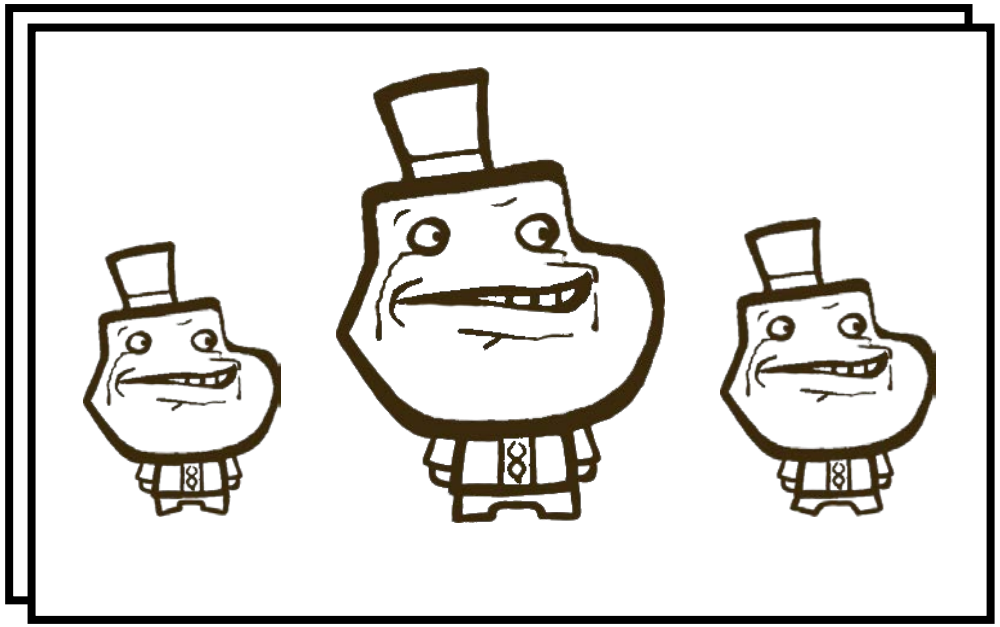
# Module —— Pygame


Pygame包含图像、声音，建立在SDL基础上，允许实时电子游戏研发而无需被低级语言（如机器语言和汇编语言）束缚。基于这样一个设想，所有需要的游戏功能和理念都（主要是图像方面）都完全简化为游戏逻辑本身，所有的资源结构都可以由高级语言提供。

使用python可以导入pygame来开发具有全部特性的游戏和多媒体软件，Pygame是极度轻便的并且可以运行在几乎所有的平台和操作系统上。

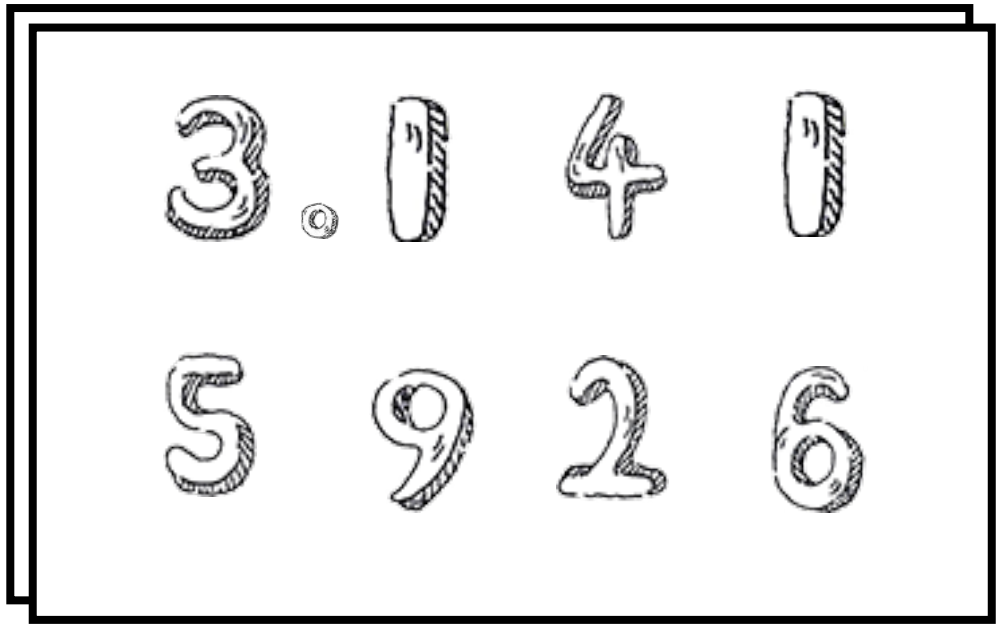

Photoshop
Autodesk
Sketchbook

# 可视化



## 角色

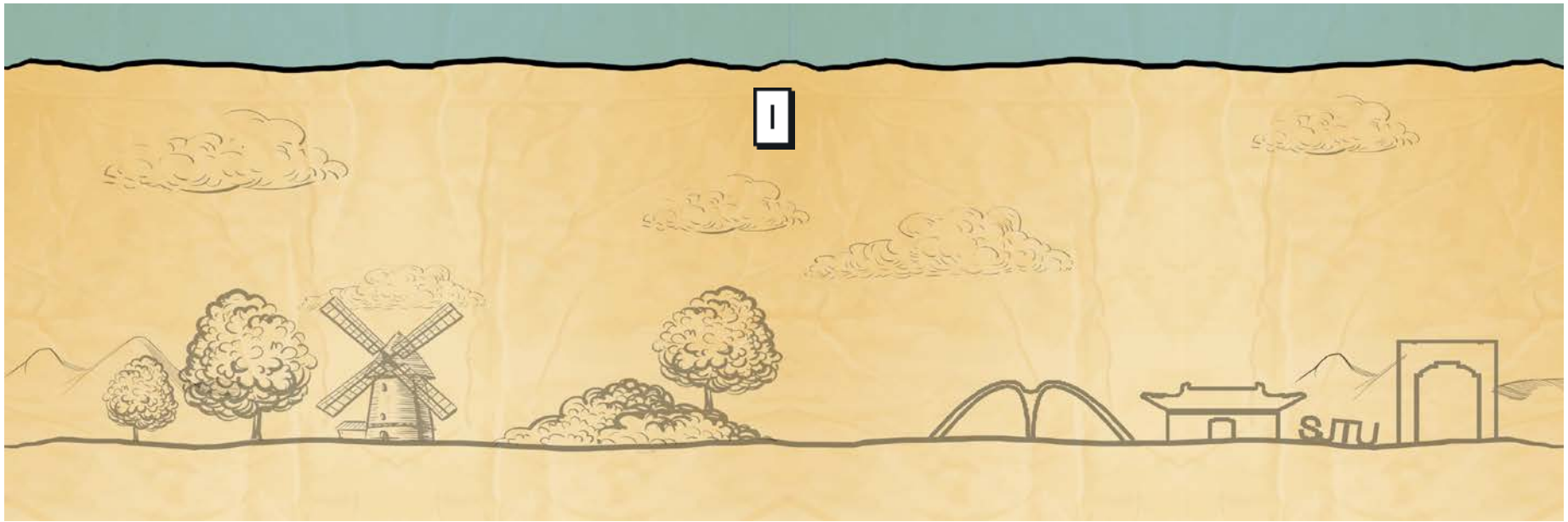
角色脚部发生微妙变化
表现出角色正在跑动的动态




## 障碍物

障碍物为数学中π的构成数字
且将根据π中数字的出现顺序依次出现

# 可视化



## 背景

牛皮纸+简笔画，塑造涂鸦感

加入交大元素

PART 04

# 代码编写

PROGRAMMING

# 基本参数

## 窗口大小

800 * 300

## 障碍物大小

0：42*55 1：27*55

2：42*55 3：42*55

4：40*56 5：35*55

6：38*55 7：41*56

8：40*55 9：41*55

## 角色尼玛大小

pose 1：70*94

pose 2：71*95

pose 3：71*94

## 更新画面时间

FPS = 50

# 定义类

## 定义滚动背景类

对背景的坐标位置

滚动状态的判断与重复做定义

```
class MyMap():

    def __init__(self, x, y):
        self.bg = pygame.image.load("image/mm2.png").convert()
        self.x = x
        self.y = y

    def map_rolling(self):
        if self.x < -790:
            self.x = 800
        else:
            self.x -= 5

    def map_update(self):
        SCREEN.blit(self.bg, (self.x, self.y))
```

# 定义类

```
class Nima():
    def __init__(self):
        # 初始化角色矩形
        self.rect = pygame.Rect(0, 0, 0, 0)
        self.jumpState = False
        self.jumpHeight = 130  # 跳跃高度
        self.lowest_y = 160  # 角色最低坐标
        self.jumpValue = 0  # 高度增加量
        self.nimaIndex = 0
        self.nimaIndexGen = cycle([0, 1, 2])
        # 加载角色
        self.nima_img = (
            pygame.image.load("image/man_obst/pose1.png").convert_alpha(),
            pygame.image.load("image/man_obst/pose2.png").convert_alpha(),
            pygame.image.load("image/man_obst/pose3.png").convert_alpha(),
        )
        self.jump_audio = pygame.mixer.Sound('audio/jump.wav')
        self.rect.size = self.nima_img[0].get_size()
        self.x = 50;  # 角色的X坐标
        self.y = self.lowest_y;  # 角色的Y坐标
        self.rect.topleft = (self.x, self.y)
```

## 定义人物类

对角色（尼玛）的坐标位置

跳跃状态、角色大小

角色形象可视化等做定义

```
def draw_nima(self):
    # 定义角色行走动图
    nimaIndex = next(self.nimaIndexGen)
    SCREEN.blit(self.nima_img[nimaIndex],
                (self.x, self.rect.y))
```

定义角色尼玛行走的动态状态及位置

# 定义类

## 定义障碍物类

对障碍物图片
进行加载

```
import random

class Obstacle():
    score = 1
    def __init__(self):
        self.rect = pygame.Rect(0, 0, 0, 0)
        # 加载障碍物图片
        self.num0 = pygame.image.load("image/man_obst/0.png").convert_alpha()
        self.num1 = pygame.image.load("image/man_obst/1.png").convert_alpha()
        self.num2 = pygame.image.load("image/man_obst/2.png").convert_alpha()
        self.num3 = pygame.image.load("image/man_obst/3.png").convert_alpha()
        self.num4 = pygame.image.load("image/man_obst/4.png").convert_alpha()
        self.num5 = pygame.image.load("image/man_obst/5.png").convert_alpha()
        self.num6 = pygame.image.load("image/man_obst/6.png").convert_alpha()
        self.num7 = pygame.image.load("image/man_obst/7.png").convert_alpha()
        self.num8 = pygame.image.load("image/man_obst/8.png").convert_alpha()
        self.num9 = pygame.image.load("image/man_obst/8.png").convert_alpha()
        # 加载分数图片
        self.numbers = (pygame.image.load('image/0.png').convert_alpha(),
                        pygame.image.load('image/1.png').convert_alpha(),
                        pygame.image.load('image/2.png').convert_alpha(),
                        pygame.image.load('image/3.png').convert_alpha(),
                        pygame.image.load('image/4.png').convert_alpha(),
                        pygame.image.load('image/5.png').convert_alpha(),
                        pygame.image.load('image/6.png').convert_alpha(),
                        pygame.image.load('image/7.png').convert_alpha(),
                        pygame.image.load('image/8.png').convert_alpha(),
                        pygame.image.load('image/9.png').convert_alpha())
        # 加载加分音效
        self.score_audio = pygame.mixer.Sound('audio/score.wav')  # 加分
```

对分数图片&音效
进行加载

# 设置障碍物出现顺序

每出现一次当前π中的第一个数字障碍物后，将调用de函数，删去π中的第一个字符。

通过if判断当前π的第一个数字，从而在障碍物中匹配相应的图片。

由此建立出按照π中数字顺序的障碍物类。

```
#定义π中数字出现的顺序
pi1 = str(math.pi)
pi = pi1[0] + pi1[2:100]
def de(a):
    return a[1:]
if pi[0] == "0":  # 如果当前π的第一个数字为0显示障碍物数字0
    self.image = self.num0
elif pi[0] == "1":
    self.image = self.num1
elif pi[0] == "2":
    self.image = self.num2
elif pi[0] == "3":
    self.image = self.num3
elif pi[0] == "4":
    self.image = self.num4
elif pi[0] == "5":
    self.image = self.num5
elif pi[0] == "6":
    self.image = self.num6
elif pi[0] == "7":
    self.image = self.num7
elif pi[0] == "8":
    self.image = self.num8
elif pi[0] == "9":
    self.image = self.num9
pi = de(pi) #障碍物变化完毕，删去当前π的第一个数字，推进至下一个障碍物顺序
```

# 背景音乐&游戏图标

游戏图标为π

游戏背景音乐采用《圆周率之歌》

```
# 设置游戏图标
icon = pygame.image.load('image/pi.jpeg')
pygame.display.set_icon(icon)
# 设置游戏背景音乐
pygame.mixer.music.load("audio/bgm.mp3")
pygame.mixer.music.play()
```

# 分数获取

```
# 获取分数
def getScore(self):
    self.score
    tmp = self.score;
    if tmp == 1:
        self.score_audio.play()  # 播放加分音乐
    self.score = 0;
    return tmp;

# 显示分数
def showScore(self, score):
    """在窗体顶部中间的位置显示分数"""
    self.scoreDigits = [int(x) for x in list(str(score))]
    totalWidth = 0
    for digit in self.scoreDigits:
        totalWidth += self.numbers[digit].get_width()
    Xoffset = (SCREENWIDTH - totalWidth) / 2
    for digit in self.scoreDigits:
        SCREEN.blit(self.numbers[digit], (Xoffset, SCREENHEIGHT * 0.1))
        Xoffset += self.numbers[digit].get_width()
```

定义getScore()以获取分数，

并在加分时播放加分音乐。

# 难度调整

获取程序开始时的时间作为初始时间，经过30s后，增加障碍物的时间增加，难度增大

```
import time

time1 = time.time()

    time2 = time.time()-time1
if time2 > 30:
    addObstacleTimer += 25  # 增加障碍物时间
if time2 <= 30:
    addObstacleTimer += 18
print(time2)
```

# 游戏结束

游戏结束时播放音频，显示游戏结束的图片并把它绘制在屏幕中央

```
# 游戏结束
def game_over():
    bump_audio = pygame.mixer.Sound('audio/bump.wav')
    bump_audio.play()
    screen_w = pygame.display.Info().current_w
    screen_h = pygame.display.Info().current_h
    over_img = pygame.image.load('image/gameover.png').convert_alpha()
    SCREEN.blit(over_img, ((screen_w - over_img.get_width()) / 2,
                                        (screen_h - over_img.get_height()) / 2))
```

# 游戏主体脚本

```
def mainGame():
    score = 0
    over = False
    global SCREEN, FPSCLOCK, word
    pygame.init()

    FPSCLOCK = pygame.time.Clock() #循环时长
    SCREEN = pygame.display.set_mode((SCREENWIDTH, SCREENHEIGHT))
    pygame.display.set_caption('Py(π)酷跑') # 设置窗口标题
    # 设置游戏图标
    icon = pygame.image.load('image/pi.jpeg')
    pygame.display.set_icon(icon)

    # 创建地图
    bg1 = MyMap(0, 0)
    bg2 = MyMap(800, 0)
    # 创建尼玛
    nima = Nima()
    addObstacleTimer = 0  # 添加障碍物的时间
    list = []  # 障碍物列表
while True:
    for event in pygame.event.get():

        if event.type == QUIT:
            exit()  # 关闭窗口
        if event.type == KEYDOWN and event.key == K_SPACE:
            if nima.rect.y >= nima.lowest_y:
                nima.jump()
                nima.jump_audio.play()
            if over == True:
                mainGame()
if over == False:

    bg1.map_update()
    bg1.map_rolling()
    bg2.map_update()
    bg2.map_rolling()
    nima.move()
    nima.draw_nima()
    # 计算障碍物间隔时间
    if addObstacleTimer >= 1300:
        r = random.randint(0, 100)
        if r > 40:
            obstacle = Obstacle()
            list.append(obstacle)
        addObstacleTimer = 0
    for i in range(len(list)):
        list[i].obstacle_move()
        list[i].draw_obstacle()
        # 判断尼玛与障碍物是否碰撞
        if pygame.sprite.collide_rect(nima, list[i]):
            over = True
            game_over()
        else:
            # 判断尼玛是否跳过障碍物
            if (list[i].rect.x + list[i].rect.width) < nima.rect.x:
                score += list[i].getScore()
        list[i].showScore(score)

        addObstacleTimer += 20
        pygame.display.update()  # 更新整个窗口
        FPSCLOCK.tick(FPS)


if __name__ == '__main__':
    mainGame()
```

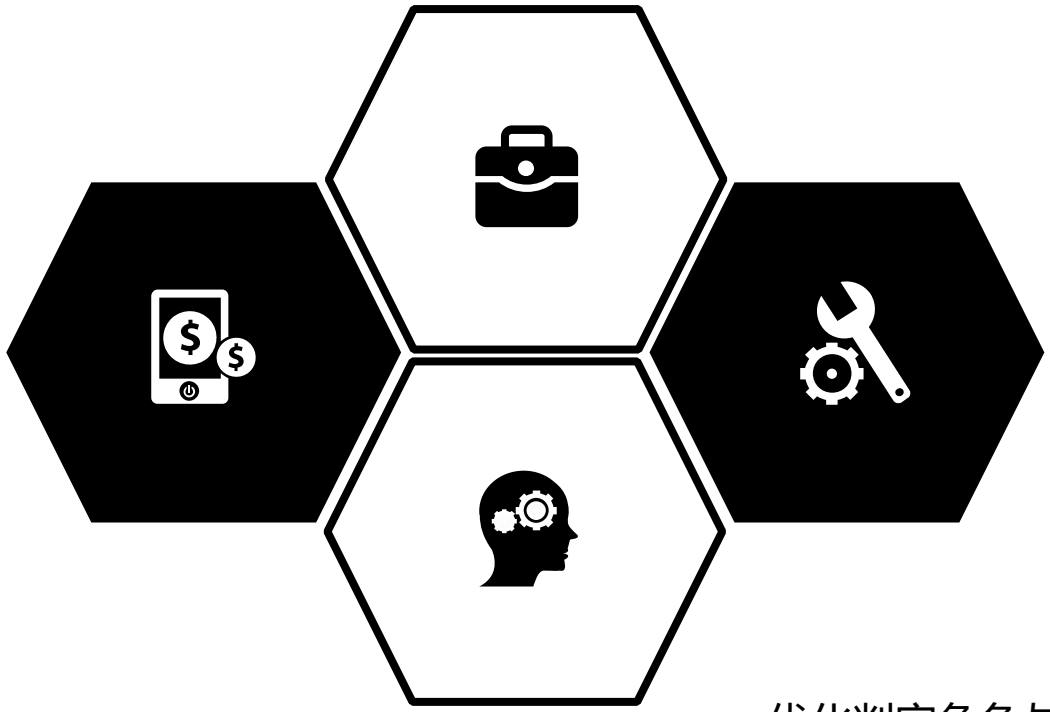
PART 05

# 总结展望

OUTLOOK&IMPROVEMENT

# 研究假设

添加加分道具
当角色的横坐标等于道具横坐标
且角色纵坐标大于等于道具纵坐标时
得分将额外增加一分



电脑外置摄像头拍摄玩家人像
轮廓的识别
实时抠像作为游戏的角色
虚拟与现实结合

如果出现的数字不是按照π的
顺序出现
则需要玩家对其进行攻击

优化判定角色与障碍物之间碰撞的算法
使得判断更加精确
避免因pgn的留白导致判断范围扩大的问题

# Py(π)酷跑

Presented by 无魏三国

**刘（备）**子欣　**孙（权）**美琪　**（周）俞**梦婕

## 感谢您的观看