

Pacman

《吃豆人》游戏设计文档



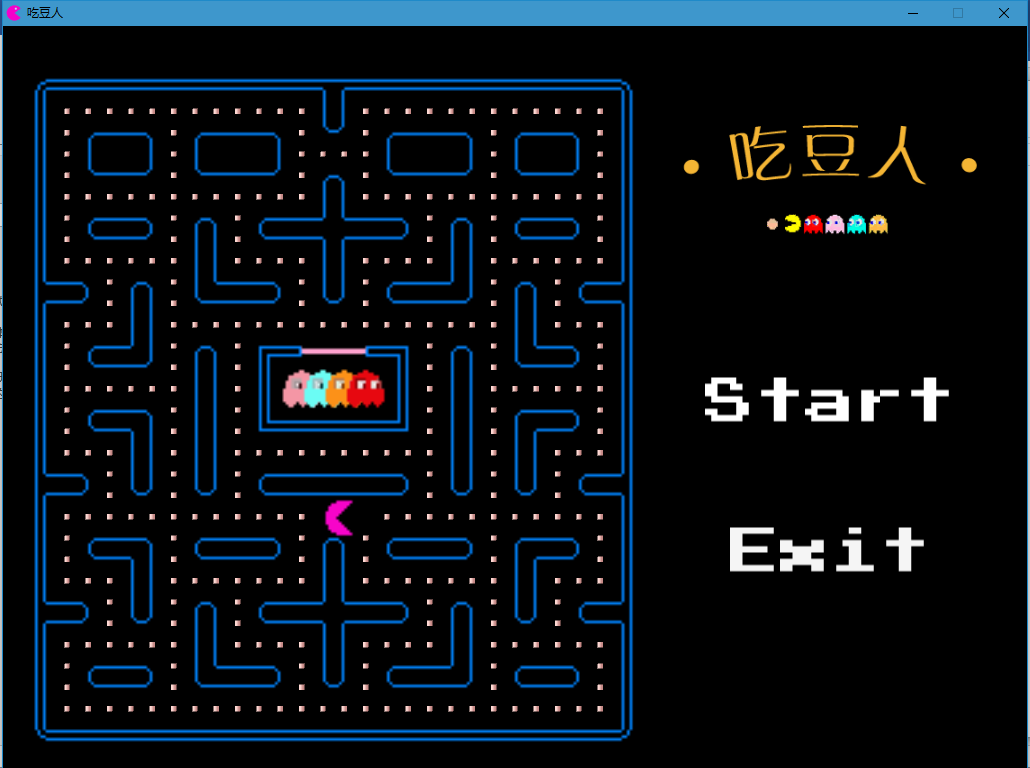
计算机科学与技术学院

校际交流1601班

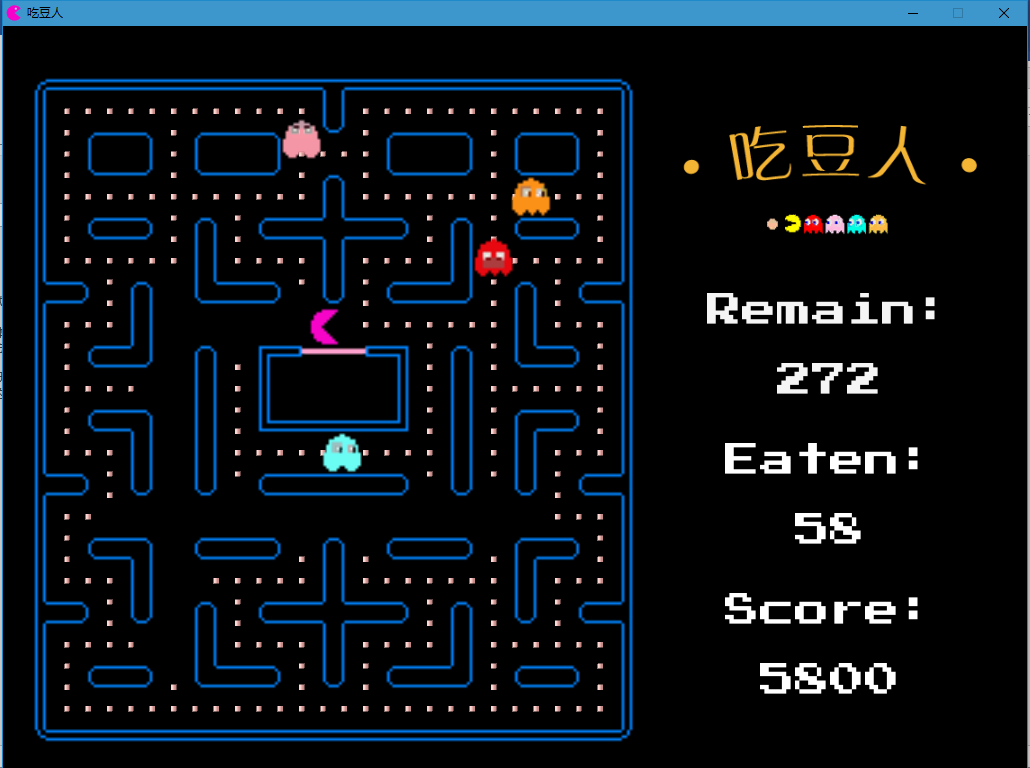
刘逸帆

U201610504

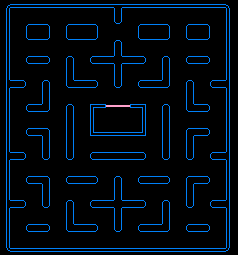
* 游戏概念
  + 吃豆人是二维的单人过关游戏，他需要玩家在躲避四只小怪兽的同时完成自己的游戏目标，考研玩家的反应能力。
* 游戏简述
  + 吃豆人游戏中，玩家需要在躲避电脑操作的四只小怪兽的同时完成自己的游戏目标，吃到地图上的所有小豆子，游戏开始界面如图。



* + 游戏中，小怪兽移动路径随机，需要主角依靠快速的移动躲避，游玩界面如图。



* 目标人群
  + 目标人群为包括儿童、青年、成年人在内的一系列玩家，主要面向儿童，可以用于锻炼儿童的反应能力、也可以用于给成年人放松等。
* 游戏元素
  + 主角：

1. 吃豆人（形象如图）：能够吃下豆子，触碰怪兽则游戏结束；吃下超级豆子后的三秒内可以吃下怪兽。
   * 敌人：
2. 小怪兽（形象如图）：运动路径随机，若触碰到没有吃下超级豆子的吃豆人则游戏结束。
   * 环境：
3. 围墙（）：有碰撞体积，角色不能够穿过围墙进行移动。
4. 小怪兽出生点（）：开始游戏时四只小怪兽均从此出现并从粉红色的门走出。小怪兽在被吃到超级豆子的吃豆人吃下后，也将回到这里。
5. 豆子（小）和超级豆子（大）（）：所有豆子被吃下都将增加100得分，超级豆子额外增加200得分。

* 玩家角色
  + 玩家操作游戏中的吃豆人躲避怪兽吃完所有豆子，取得游戏的胜利。
* 游戏世界
  + 游戏部分截图如下。

1. 玩家吃到超级豆子后将敌人吃下并送回出生点：



1. 玩家吃完所有豆子后取得游戏胜利：



1. 玩家触碰到小怪兽后从地图上消失，游戏失败：



* + 视觉上：地图十分简明清晰，不需要任何的提示就能让玩家明白需要如何游戏；小怪兽与玩家的颜色各不相同，易于区分且丰富了界面；同时每个角色都通过组件实现了动态形象，十分可爱。游戏整体画面清晰，逻辑简单。
  + 听觉上：简单循环的背景音乐，但也烘托出了应有的游戏感。
* 游戏机制
  + 角色操作：键盘ADWS或左右上下控制吃豆人的移动，功能分别为左移、右移、上移、下移。
  + 游戏机制：玩家吃下豆子会增加计分板上的100积分，吃到小怪兽会使游戏结束，其中小怪兽会随机移动。每10秒钟地图上会随机出现一个超级豆子（最多同时存在一个），玩家吃下后可短时间（3s）内使小怪兽暂停移动并可以将其吃下，其中吃下超级豆子记分板上积分增加300，吃下小怪兽积分增加500。
  + 游戏结果：地图上所有豆子被吃尽则游戏获胜，触碰到小怪兽则游戏失败。
* 引用
  + 网上搜索的Pacman游戏-音乐与图片素材

部分代码设计介绍：

* GameManager的部分代码介绍：

1. 分别使用了Awake、Start与Updata函数，在Awake函数中完成了对地图等对象的实例化，在Start中冻结了所有对象的移动，在Updata函数中判断游戏是否结束并更新积分版的内容。

//Awake()在脚本对象实例化时被调用的

private void Awake()

{

\_instance = this;

Screen.SetResolution(1024, 768, false);

int tempCount = rawIndex.Count;

for (int i = 0; i < tempCount; i++)

{

int tempIndex = Random.Range(0, rawIndex.Count);

usingIndex.Add(rawIndex[tempIndex]);

rawIndex.RemoveAt(tempIndex);

}

foreach (Transform t in GameObject.Find("Maze").transform)

{

pacdotGos.Add(t.gameObject);

}

pacdotNum = GameObject.Find("Maze").transform.childCount;

}

//Start()在对象的第一帧时被调用的，而且在Update()之前

private void Start()

{

SetGameState(false);

}

private void Update()

{

if (nowEat == pacdotNum && pacman.GetComponent<PacmanMove>().enabled != false)

{//若吃完所有豆子则提示游戏胜利并锁定所有角色的移动

gamePanel.SetActive(false);

Instantiate(winPrefab);

StopAllCoroutines();

SetGameState(false);

}

if (nowEat == pacdotNum)

{//豆子已被吃完且已提示游戏胜利时，输入任意键回到游戏开始界面

if (Input.anyKeyDown)

{

SceneManager.LoadScene(0);

}

}

if (gamePanel.activeInHierarchy)

{

remainText.text = "Remain:\n\n" + (pacdotNum - nowEat);

nowText.text = "Eaten:\n\n" + nowEat;

scoreText.text = "Score:\n\n" + score;

}

}

1. 在设计开始游戏倒计时的过程中，使用了协程来加快整个游戏的准备效率；并使用了委托函数Invoke，用来延迟调用函数使产生超级豆子。

IEnumerator PlayStartCountDown()

{

GameObject go = Instantiate(startCountDownPrefab);

yield return new WaitForSeconds(4f);//暂停协程 4s后执行后续代码(游戏开始倒计时)

Destroy(go);

SetGameState(true);//解锁所有角色的运动

Invoke("CreateSuperPacdot", 10f);//委托 延时调用函数产生特殊豆子

gamePanel.SetActive(true);

GetComponent<AudioSource>().Play();

}

同时每次吃完超级豆子后产生新的超级豆子也使用了委托机制。

//吃到特殊道具(大豆子)时

public void OnEatSuperPacdot()

{

score += 200;

Invoke("CreateSuperPacdot", 10f);//委托 延时调用函数产生特殊豆子

isSuperPacman = true;

FreezeEnemy();

StartCoroutine(RecoveryEnemy());

}

* PacmanMove主角移动的部分代码介绍：

由于地图的设计比较简单的原因，没有太多情形需要考虑，在给地图与吃豆人都挂上碰撞体积后，只需要根据输入使吃豆人移动。

private void FixedUpdate()

{

//插值得到要移动到dest位置的下一次移动坐标

Vector2 temp = Vector2.MoveTowards(transform.position, dest, speed);

//通过刚体来设置物体的位置

GetComponent<Rigidbody2D>().MovePosition(temp);

if (true)

{

if (Input.GetKey(KeyCode.UpArrow) || Input.GetKey(KeyCode.W))

{

dest = (Vector2)transform.position + Vector2.up;

}

if (Input.GetKey(KeyCode.DownArrow) || Input.GetKey(KeyCode.S))

{

dest = (Vector2)transform.position + Vector2.down;

}

if (Input.GetKey(KeyCode.LeftArrow) || Input.GetKey(KeyCode.A))

{

dest = (Vector2)transform.position + Vector2.left;

}

if (Input.GetKey(KeyCode.RightArrow) || Input.GetKey(KeyCode.D))

{

dest = (Vector2)transform.position + Vector2.right;

}

//获取移动方向

Vector2 dir = dest - (Vector2)transform.position;

//把获取到的移动方向设置给动画状态机

GetComponent<Animator>().SetFloat("DirX", dir.x);

GetComponent<Animator>().SetFloat("DirY", dir.y);

}

}

其中最后两条语句将移动方向提交给状态机是为了使主角模型转向，否则游玩过程将会不自然。