毕业设计(论文)说明书

题　目：**HTML5技术在Web游戏开发中的应用与探索**
系　别：信息工程系
专　业：计算机科学与技术
学生姓名：曾铭
学　号：0751210126
指导教师：赵莹莹
职　称：讲师

题目类型：　理论研究□　实验研究□　工程设计□　工程技术研究□　软件开发√

2011年6月1日

摘 要

HTML5是下一代的Web标准， Web Socket、Canvas等技术的引入为胖客户端网页开发带来的全新的发展空间。六轴陀螺仪等传感器在iOS设备中的普遍应用，也为游戏带来全新的控制方式。

本课题对HTML5进行技术探索和研究。以Web游戏开发为研究对象，以桌面浏览器iOS设备游戏终端，使用HTML5技术、（主要是其中的Web Socket 、Canvas 设计网络游戏架构并实现了游戏Demo： Run Zombies Run。

课题侧重于Web游戏开发中的 Canvas 和 Web Socket 等技术的架构设计和实现。在服务器端和客户端分别主要使用Ruby和JavaScript语言作为开发语言，实现HTML5双人互动网页游戏（其中一人使用iOS设备作为体感控制器）。

本课题研究与实践成果为基于HTML5的Web游戏开发架构等带来实际参考，具有较强的实践推广价值。

关键词：HTML5；Web Socket；Canvas；体感控制；Web游戏开发

Abstract:

Key words: HTML5; Canvas; Web Socket; Somatic Sensation; Web Game Development;

目　录

引言

Web游戏因其基于浏览器运行便利、无需安装、安全性高等特性深受轻度游戏用户的欢迎。纵观各大社交网站，无论是国外的Facebook还是国内的人人网、腾讯朋友等网站，基于Flash的Web游戏开发如火如荼。另一方面HTML5的发展方兴正艾，如何在Web游戏中应用HTML5技术已经成为诸多游戏开发者的试验田。

本课题在Web游戏开发中，使用Web Socket和Canvas两种技术，探索实现了小型网页游戏Run Zombies Run。

1. 课题背景与意义

1. 行业背景

目前 HTML5 标准尚未完全确立，但主流浏览器厂商均大力支持并推广HTML5标准，并因为 HTML5 标准良好的向上兼容性，其诸多特性已经进入实用阶段。根据优雅退化的设计原则，Google、Apple、HP、阿里巴巴、腾讯等企业已经在HTML5方面做了诸多尝试，Gmail、Gtalk、Web QQ等成熟的商业产品也已经使用多种HTML5技术来为用户带来更好的使用体验。

2. HTML5背景

1.总体介绍

广

草

H

Summary

Calculation of support for currently selected criteria

	IE	Firefox	Safari	Chrome	iOS Safari	Android Browser	
Two versions back	7.0: 0%	3.5: 48%	3.2: 0%	9.0: 86%	3.2: 36%	2.1: 43%	
Previous version	8.0: 34%	3.6: 63%	4.0: 36%	10.0: 86%	4.0-4.1: 57%	2.2: 50%	
Current	9.0: 41%	4.0: 82%	5.0: 71%	11.0: 93%	4.2-4.3: 64%	2.3: 50%	3.0: 64%
Near future	9.0: 41%	5.0: 82%	5.0: 71%	12.0: 93%			
Farther future	10.0: 41%	6.0: 91%	6.0: 79%	13.0: 93%			

图1-1 、目前各浏览器对HTML5的支持情况

2.Web Socket 介绍

为什么要有Web Socket呢？这就要先看看没有Web Socket的时代。

http 协议数据的传输模型是请求响应模型。基本的应用方式就是客户端（浏览器）发送请求，服务器收到请求后做出相应响应，然后客户端（浏览器）处理服务器发来的响应信息。这种数据传输模型用于网页浏览没有问题，而对于富客户端的Web应用则极其受限。

开发者需要一种数据传输方式，既能让客户端主动给服务器端发送信息，服务器端也能够“主动”给客户端发送信息。即实现客户端与服务器端的双向通信。

以往变通实现这种双向通信有两种方式，一种是服务器轮询，即客户端使用Ajax周期性的向服务器端发送请求，如果服务器端需要向客户端发送信息，则在响应中加入内容，否则返回空响应（仍然要响应）；另一种是长连接方式，即客户端使用Ajax向服务器端发送请求后，服务器并不立即响应，等服务器端需要发送数据是再响应，然后客户端继续发送请求，维持长连接。

这两种方式都存在实现复杂，无法实时数据传输，冗余数据多，传输效率差，服务器压力大等问题。但由于没有其它选择，仍被广泛的使用与富客户端的Web开发中。

这里需要强调另一个名词Ajax。其全称是 Asynchronous JavaScript and XML（异步的JavaScript 和 XML）。它并非是一个技术，而是JavaScript，XML，XMLHttpRequest，CSS等一系列技术有机结合的总成。主要可以不刷新页面实现客户端与服务器端的数据交互。要注意的是Ajax网络数据传输使用的仍然是已有的请求响应模型，并没有给服务器端和客户端带来持久的双向通信。

Web Socket 是 HTML5 引入的一项重要特性。在此之前，使用 Web 是无法直接实现持久的双向通讯的。伴随着 Ajax 的普及，也有相关的技术来变相的实现 B/S 持久双向通信，如服务器轮询，长连接等。限于技术本身限制，这些实现方式都存在着实现复杂，冗余信息量大等问题。

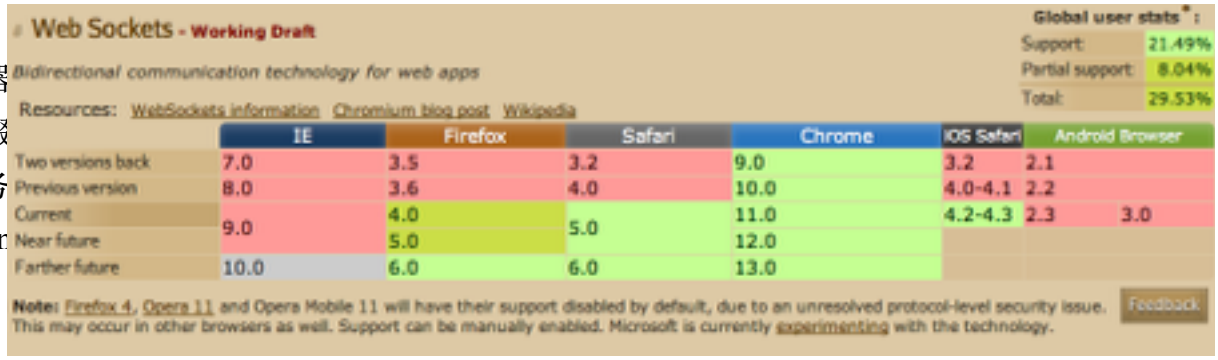

图1-2 、目前各浏览器对Web Socket的支持情况

3.Canvas 介绍

Canvas 是 HTML5 新引入的画板标签。简单来说，Canvas 可以完成之前 HTML 标准无法完成的复杂绘图任务，如游戏，视频等。

HTML5 之前，想通过 HTML 自身实现动画是一件很困难的事。它唯一支持的就是通过 JavaScript 语句实现 DOM 节点的简单动画。由于 DOM 自身的限制，这种方式无法实现游戏开发中复杂的动画效果（如 CSS3 之前并不支持 DOM 节点的旋转等）。

在当时的环境下，进行 Web 游戏开发的“几乎唯一选择”就是 Flash。（Silverlight 因市场占有率问题并不受开发者重视）短期来看，这没有什么。但因为 Flash 是是由标准，不同平台的实现依靠唯一一家公司 Adobe，而其性能，安全性等一直备受诟病。Mac，Linux 下 Flash 的性能也一直差强人意。苹果公司推出 iOS 设备后，基于其平台统一性及性能原因的考量，明确表态不会支持 Flash。这更成为 Flash 这一封闭标准步入黄昏的标志性事件。

HTML5出现之后，也部分解决了这些问题。更丰富的Web 效果，跨平台的Web应用

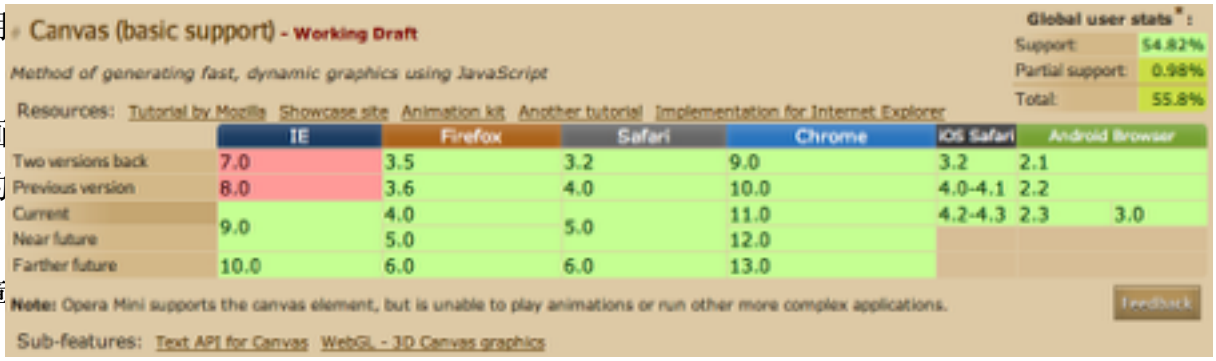

图1-2 、目前各浏览器对Canvas的支持情况

3. 课题的意义

HTML5的发展尚属初级阶段，各方面架构、类库并未成型。Web游戏开发中HTML5的应用也尚未形成成熟的设计模式。

本课题通过对一个小型游戏的架构设计、及各个技术的整合运用，提出一套解决方案，实现了一个小型的Run Zombies Run游戏。项目设计过程、开发文档、代码完全开源，项目托管在Github（https://github.com/merlyle/run-zombie-run ），分享给广大Web游戏开发爱好者们。

2. 总体设计

4. 核心技术分析及选取

既然项目是基于HTML5技术的，HTML5有诸多新内容一个项目的使用要对其进行斟酌选用。本项目主要使用如下HTML5技术：

(1) Canvas API：进行二维绘图；

(2) HTML5音频：为游戏提供音频支持；

(3) CSS3:实现更丰富的 DOM 显示效果；

(4) Web Socket：实现浏览器与服务器全双工通信的基础；

(5) iOS设备传感器API：使用iPad作为体感控制器。

对于客户端实现，使用的必然是JavaScript语言；对于服务器端，则可根据情况选用不同语言的实现（接口是相同的）。服务器端，目前php，java，ruby均有各自实现。笔者根据项目情况，选用ruby作为服务器端开发语言，同时使用eventmachine和em-Websocket 两个Ruby Gem来实现Zelda服务器。

2. 游戏模式设计

4.移动模式

因为是探索性质的技术学习，所以并没有在游戏设计上花费太多功夫。刚巧笔者实习公司在做一个僵尸围攻玩家的游戏，于是借鉴其游戏方式做出如下构想：

(1) 一个打开即能玩的Web 游戏，玩家可通过键盘控制自己的 player，控制方式有：转向，速度调整，攻击面前扇形区域；
(2) 画面中有若干 zombie，每个zombie 都有自己的视野。当视野中有player时，则向player移动，攻击 player。当视野中没有player时，zombie群落模拟人工生命，自主移动。
(3) player和zombie都有自己的血槽，当zombie碰到player时，player损失一定血量，当player发动攻击时，面前扇形内zombie损失一定血量。
(4) 可双人联网游戏，另一玩家可以使用 iOS 设备，打开控制器网页加入到游戏中，控制另一个player，实现两个player合作躲避zombie攻击同时攻击zombie。
(5) 控制器网页在iOS设备的mobile safari 中运行，加入体感的方式来控制游戏。
(6) 实现两个player 的聊天功能。
(7) 游戏有计分机制，攻击有动画效果，有音效，可暂停，
(8) 游戏有 debug 信息，便于调试。

2.攻击模式

当 zombie 视野中有 player 时，设计 zombie 主动向 player 运动，攻击玩家。当 zombie 视野中没有 player 时，这时zombie 的运动处理是个难题，不作处理或随机运动都可能有运动轨迹怪异，重叠碰撞等问题发生。

游戏中针对这个问题，参考柏兹群聚模型，实现了 zombie 运动智能化。

3.Zombies 人工生命模式

准确地说柏兹模型并非 AI （人工智能）模型，而是属于人工生命的范畴。人工生命侧重于通过人为设计逻辑，使设计群体体现出生命特征。

柏兹模型地基本原则非常简单：

(1) 避免碰撞（Collision Avoidance）

首先临近生命维持一个最小距离，当小于这一最小距离时，相互远离避免碰撞。

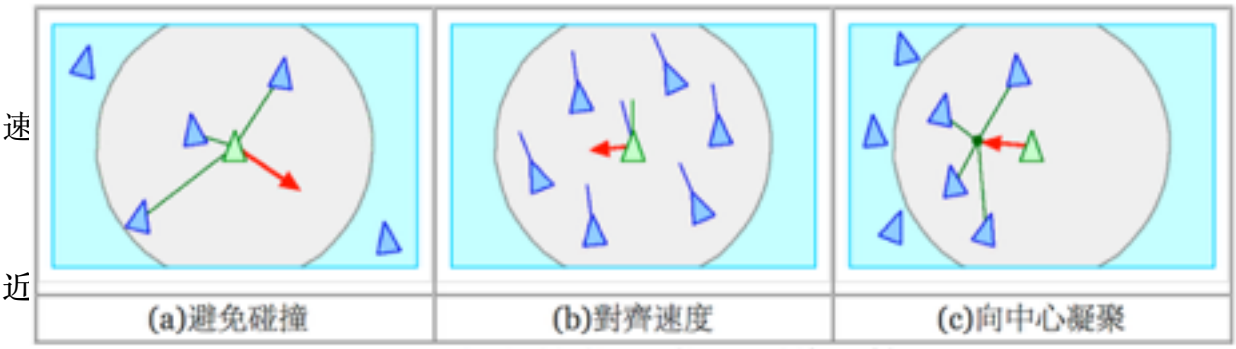

图2-1、柏兹群聚模型示意图

图片摘自：混沌边缘的艺术，人工生命（参见附录）http://redbug0314.blogspot.com/2009/06/blog-post.html

以上三条规则便是柏兹模型地全部，经观察，通过柏兹模型设置运动方式的个体，有类似生命群体地运动迹象，如鸟群的飞翔，鱼群的洄游等。本项目中，将对于非攻击状态 zombie 定义柏兹模型来确保其运动的合理性。

3. 基于 Web Socket的通信系统详细设计及实现

3. Zelda服务器平台的详细设计

开始来做我们的游戏吧！

可是，怎么做一个网络Web游戏呢?

在进行flying zombie 设计时可将其作为一个单机Web游戏来设计。只需将其控制方式，聊天功能等封装对应的接口，本地键盘玩家的控制通过接口的方式来控制player，后续网络控制器玩家也通过服务器转发控制及聊天信息，这些信息调用相应接口就能够实现网络游戏的功能了。

所以，问题可以转化为：如何做一个单机Web游戏?

根据上面的分析，共有三个部分需要完成：

(1) flying zombie，游戏画面的显示页面（取题来源于游戏画面中主要是移动中的僵尸）。这个页面中的动画效果主要是使用Canvas绘图技术和JavaScript操作DOM节点技术来实现动画效果。所用语言是JavaScript，CSS，HTML。

(2) wind，游戏iOS设备控制器页面（取题来源于控制器页面wind驱动了flying zombie 页面中player的运动）。这个页面主要是使用iOS设备传感器技术实现体感控制，使用CSS3实现控制器布局等。所用语言依旧是JavaScript，CSS，

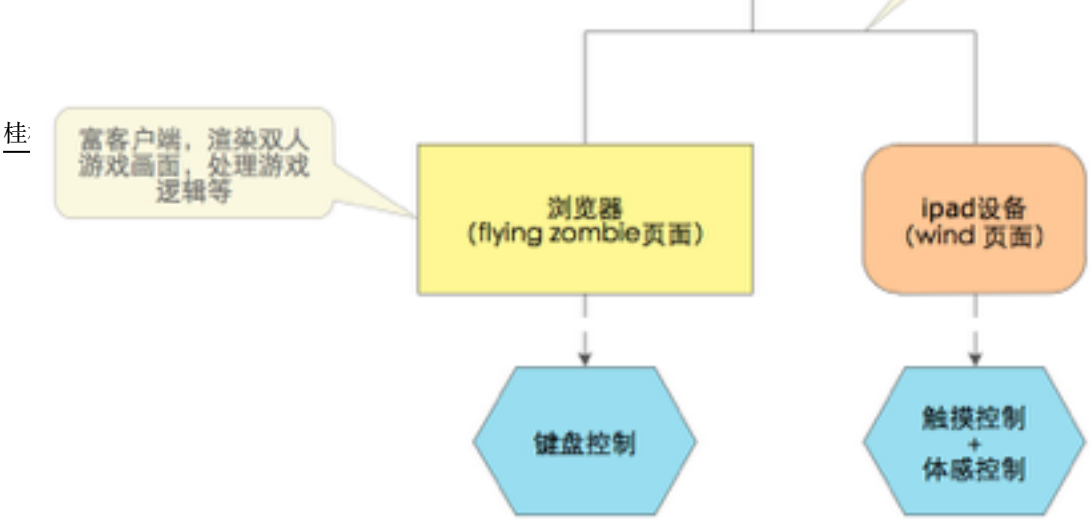

图3-1、Run Zombies Run游戏方式示意图

架构有了，模型有了，算法也有了，实现游戏便不是困难的事了。下面一步一步进行设计与实现。

2. 消息接口设计

消息接口的设计目的是定义发送方与接收方均可识别的消息格式。实践项目采用多层封装的方式来定义接口。具体定义方式如下：

客户端往服务器端发送的消息是一个字符串，其中第一个字符是接收方识别码，第二个字符到第四个字符是消息类型识别码，后面的内容为消息内容（此处需注意的是个别消息类型中消息内容仍有多层封装，接收方处理时会对消息内容再次解构。如聊天类型的消息正文中同时包含聊天发送方和聊天内容）。

整体消息接口设计可用下表表示。

表3-1、消息结构设计

接收方识别码	消息类型识别码	消息内容
0	1 ~ 3	4 ~ end

对消息结构举例，有一个消息整体是如下内容：

“fctlUpV”

zelda收到这个消息后，根据接收方识别码f把这个消息转发给剩下的内容转发给flying zombie 页面。然后flying zombie页面收到余下的信息“ctlUpV”，flying zombie 页面的根据控制信息 ctl 将消息指派给控制信息处理函数，控制信息处理函数根据UpV这个命令，驱动wPlayer加速。这样就完成了整个消息传输分发处理的过程。

4.接收方识别码设计

接收方识别码的定义用下表示：

表3-2、接收方识别码设计表

接收方识别码	名称	意义	举例
f	flying zombie 信息	需要转发给flying zombie页面的信息	fctlKeepV
w	wind 信息	需要转发给wind页面的信息	
z	zelda信息	需要zelda进行处理的信息	
c	chat 信息	需要转发给chat频道的信息	cmsgsomeone:我来了

2.接收方消息类型设计

表3-3、消息类型设计表

消息类型识别码	名称	意义	举例
ctl	控制信息类型	针对player 的控制信息消息类型	fctlFire
lik	连接信息类型	需要zelda处理的连接消息类型，将根据消息内容处理连接。	
cht	chat 系统消息类型	指示该消息为chat系统消息	cchtJoin
msg	chat 消息类型	chat聊天是普通消息类型	cmsgboy:我来聊天

3.接收方消息内容设计

表3-4、zelda lik 类型消息内容设计

消息内容	名称	需处理的动作

FlyInit	Flying zombie 客户端初始化	持有 flying zombie client Web Socket 连接；订阅 chat channel
WindInit	Wind 客户端初始化	持有 wind zombie client Web Socket 连接；订阅 chat channel

表3-5、flying zombie ctl 类型消息内容设计

消息内容	名称	需处理的动作
UpV	加速	调整速率加速度为1
DownV	减速	调整速率加速度为-1
KeepV	速度保持	调整速率加速度为0
X	X键按下	触发X键按下事件
Xend	X键松开	触发X键松开事件
Y	Y键按下	触发Y键按下事件
Yend	Y键松开	触发Y键松开事件
A	A键按下	触发A键按下事件
Aend	A键松开	触发A键松开事件
B	B键按下	触发B键按下事件
Bend	B键松开	触发B键松开事件
F1	F1键按下	触发F1键按下事件
F2	F2键按下	触发F2键按下事件
F3	F3键按下	触发F3键按下事件
F4	F4键按下	触发F4键按下事件
F5	F5键按下	触发F5键按下事件
F6	F6键按下	触发F6键按下事件

表3-6、聊天室 cht 类型消息内容设计

消息内容	名称	需处理的动作

Join	加入聊天室	有人加入聊天室，根据flying zombie和wind页面的自身情况进行提示

3. 通信系统各模块的具体设计与实现

4.Zelda 服务器端的设计与实现

根据项目运行环境，Web Socket选用的是Ruby语言的一个Gem库[1]：em-Websocket。通过em-Websocket，可方便的启动Web Socket服务器，只需定义启动信息和处理Web Socket回调函数即可。

flying zombie 页面和wind页面发送数据有很大的不同，为了分离回调函数的处理过程，zelda需要启动两个Web Socket 服务器，一个用于连接flying zombie页面，一个用于连接wind页面。

目前规定服务器均使用本机IP，两个Web Socket服务器分别使用3101和3102端口。这样服务器的基本信息就有了，下一步看如何定义Web Socket 的四个回调函数。

回调函数onOpen在Web Socket连接建立时调用。设计此时zelda向客户端发送服务器状态，客户端收到服务器状态后主动提交客户端身份信息，再由zelda在onMessage函数中验证。

回调函数onMessage当Web Socket持久连接传输数据时发生调用。zelda接收消息，根据接收方识别码，对消息进行拆分转发。当接收方识别码是z时，代表是需要zelda进行处理的信息，此时zelda不对消息进行转发，而是根据消息类型对消息进行处理。

这里需要强调的是如何持有Web Socket，在连接之后便于后续对多个客户端的消息分发。这里要用到另一个Gem：eventmachine。eventmachine中定义了channel（频道）类，当一个Web Socket 连接subscribe（订阅）一个channel后，以后每次这个channel收到信息后，就会调用subscriber（订阅者）订阅时指定的回调函数。本项目的两天功能，就是定义了chat channel，当客户端连接后，自动订阅该频道。之后zelda无论收到谁发送的聊天信息，只需转发给chat channel，chat channel就会把聊天信息分发给订阅者，从而实现类似聊天室的功能。

回调函数onClose当Web Socket连接断开是发生调用。此时需要处理清理连接变量，unsubscribe（取消订阅）相应channel等任务。

回调函数onError则是当Web Socket连接发生错误时调用。这个在正常的使用过程中不会发生，只需在函数中定义出错提示信息即可。

2.Flying Zombie 浏览器端的设计与实现

客户端 Web Socket接口和服务器端大体一致，只是用JavaScript来定义相应的回调函数。

onOpen回调函数，连接成功后向zelda发送zlikFlyInit消息，申请完成flying zombie连接初始化工作；

onMessage回调函数，根据上一章的消息接口设计，完成sys、ctl等类型消息的分发工作。根据收到消息类型的不同，ctl消息分发给handleCtlMsg函数，sys消息分发给handleSysMsg函数，聊天信息分发给handleMsgMsg函数，再由不同的函数处理相应信息。

onClose回调函数，在连接断开时调用。主要用来重置连接使得配置信息。此时注意是处于客户端环境，则在断开时进行相应提示，同时尝试重新连接。

onError回调函数，在连接出错时调用，正常连接时不会调用此函数，当调用时提示用户连接出错即可（可忽略）。

为了方便消息的分发，onMessage函数会将收到的消息封装成MsgStruct。根据此对象，可方便的获取消息类型识别码和消息类型信息。后续如果需要对消息的封装进行调整，也只需要调整MsgStruct的封装而已。

这里需要强调的是聊天功能中msg类型消息的封装，由于聊天信息中需要包含用户名信息，所以设计消息内容为“用户名”+“：”+“聊天内容”的格式，handleMsgMsg函数接收聊天消息后会拆分消息内容，做相应处理。

3.Wind控制端的设计与实现

wind页面 Web Socket 相应的内容整体与 flying zombie页面类似。不同之处在于 wind 页面是控制端，有更多向 zelda 发送消息的情况。

4.聊天模块的实现

由于聊天是flying zombie页面和wind页面都有的功能，所以重用了大部分代码，只是在显示部分，flying zombie 根据player 的位置动态显示聊天内容，而wind则是所有聊天内容显示在聊天框内。

4. Flying Zombie浏览器端动画的详细设计及实现

1. Canvas 绘图

Canvas 是 HTML5引入的允许动态渲染绘图得新标签。对Canvas认识的一大误区就

是认为其直接支持动画效果。

其实Canvas只提供了绘制基本形状（矩形，圆，直线，贝塞尔曲线，路径）和图像得功能，而这些都是静态图案，并非动画。所以使用Canvas进行游戏开发时实现动画效果其实是绘制静态图案，然后周期性的调用事件处理逻辑和绘图函数，在事件处理逻辑中对游戏中实体方向，速度，位移，渲染方式等属性进行调整，从而实现动画效果。

如GlobalManager类的start方法中定义事件处理和画面渲染回调函数：

```
start: function() {
    /** 建立事件循环和绘图循环 */
    this.eventIntervalId = setInterval(this.eventLoop, CPS_TIME);
    this.drawIntervalId = setInterval(this.drawLoop, FPS_TIME);
},
```

Player类中绘制自身的代码：

```
cx.beginPath();
cx.arc(this.x, this.y,
    this.radius, this.dir/10000, this.dir/10000+Math.PI*2);
cx.lineTo(this.x, this.y);
cx.closePath();
cx.fill();
cx.stroke();
```

其中，cx为Canvas绘板的2D绘图对象：

```
fCanvas.getContext("2d");
```

2. DOM 动画效果的实现

DOM即文本对象模型，CSS则是层叠样式表，通过CSS属性来指定DOM节点的显示位置，背景图，边框效果，阴影，旋转角度等信息。

DOM实现动画效果其实还是编写JavaScript代码，在代码中周期性控制DOM节点的CSS属性，从而达到动画的效果。如果手写代码实现这全部的效果相对复杂，本项目选用了两个JavaScript 库来编写相应代码：

(1) jQuery

(2) 动画库

3. Canvas 与 DOM 动画的结合应用

Canvas 拥有相对自由的绘图方式，方便实现丰富的自定义效果，而代码实现相对

复杂。DOM 动画则拥有更多的 API 可供调用，可方便的现实 CSS3 特效和添加事件侦听等。

面对二者的不同，开发这可考虑针对不同的内容的动画使用不同的实现方式。如游戏中 player，zombie 等复杂对象使用 Canvas 绘图的方式绘制到画板上，而聊天气泡，攻击特效，玩家得分等动画使用 DOM动画的方式来实现。

在此有一点需注意，(截止2011年5月)目前多数浏览器实现了对CSS3硬件加速的支持，而还没有浏览器实现 Canvas 硬件加速，所以 Canvas 绘图相对性能较低（目前没有硬件加速，已经满足基本的游戏应用需求）。但相信按照目前 HTML5的发展势头，这个问题将很快解决。所以这点不是笔者实现 Web 动画时技术选择的考量。

多种这计大式过

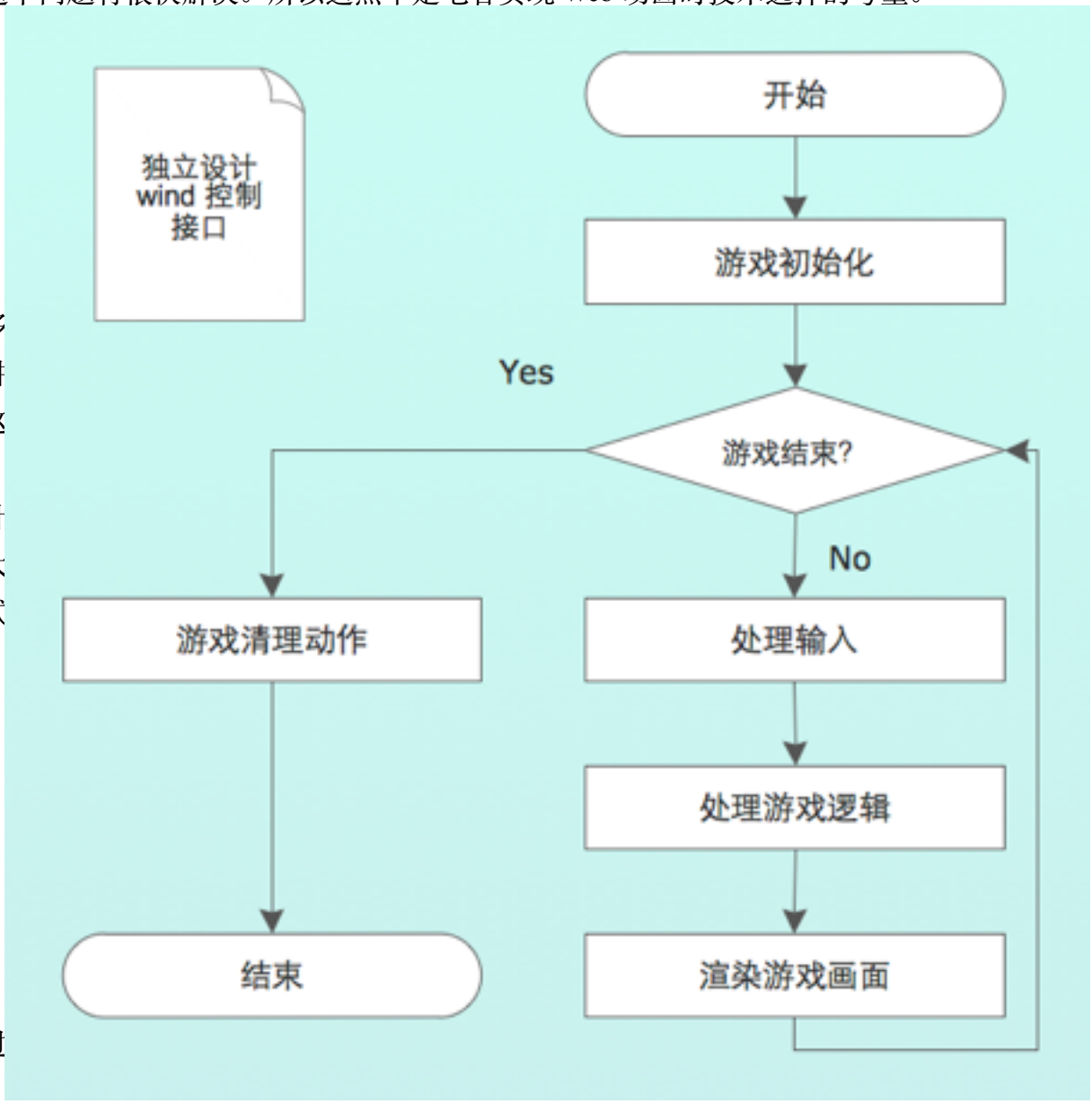

图4-1、flying zombie 页面游戏架构设计示意图：

2. Global Manager

global manager 的作用就是封装游戏的五个步骤，具体实现方式是：

(1) 通过初始化函数，实现游戏开始时的初始化功能；

(2) 监听玩家键盘输入，分发处理输入事件；

(3) 周期性的回调事件处理函数，实现实体逻辑处理；

(4) 周期性的回调绘图函数，在Canvas上绘制场景，zombie，player等内容；

(5) 清理函数。实现游戏结束时的清理动作。

总的来说，global manger 就是实现全局的游戏控制，封装游戏架构逻辑。

3. Model

global manager 已经对游戏架构进行了封装。而model则是对相应实体进行的封装。将每个实体对应的属性信息，运动逻辑，渲染方式等信息封装其中。

(1) role 类

游戏角色实体基类。定义角色必备的坐标(x, y)，方向(dir)，速度(v)，加速度(a)。实现 eventLoop 方法，当调用 eventLoop 方法时，实现 role 的转向与移动。

(2) player 类

指代游戏中玩家控制的角色，继承自 role 类。在role基础上添加存活状态（isLive），并用 Canvas绘图方式实现了 player 的绘图。

(3) zombie 类

指代游戏中的 zombie，继承自 role 类。在 role 基础上添加存活状态（isLive), 用 Canvas 绘图方式实现 zombie 的绘图。另外重要的一点是，实现游戏中 zombie 运动的 AI，在 zombie 视野中有 player 的情况下， zombie 会攻击 player，主动向 player 移动。当 zombie 视野中没有 player 时，则实现了一套 柏兹运动模型， zombie 根据此AI进行运动。（后续有详细介绍）

(4) msgStruct 类

消息类。定义消息id，消息类型（kind）和消息内容（content）

5. wind 控制器端的详细设计与实现

5. 交互界面设计及实现

wind页面的作用是将iPad作为游戏的控制手柄，所以设计时尽量使用仿手柄的界面。

网页整体使用Div + Table嵌套地方式实现上中下三栏，中栏垂直居中的布局。

图5-1、iPad设备上的wind界面截图

mobile safari是iOS设备的浏览器，其有些特性根据移动设备的情况进行设定，而有些并不是我们所需要的，甚至某些特性会严重阻碍用户使用此“手柄”。比如iPad会根据设备方向自动旋转屏幕，手指长按屏幕后会出现放大镜以帮助用户选择，多点触摸时开合手指缩放屏幕等。对于wind页面，要回避这些特性，所以要屏蔽mobile safari的默认行为。

其实屏蔽mobile safari的默认行为并不复杂，一是提供meta信息，告诉mobile safari将这个页面当作一个视口来处理，如：

<meta id="viewport" name="viewport" content="width=768; initial-scale=1.0; maximum-scale=1.0; user-scalable=0;" />

二是监听浏览器的touchstart，touchmove和touchend事件，顾名思义，这三个事件是在用户点击页面、移动以及离开页面时触发的，对其屏蔽浏览器默认行为即可：e.preventDefault();。这里还需注意的是，屏蔽这三个事件后，浏览器的click事件也不会响应，使用jQuery时绑定click事件也就不会有任何效果。所有的点击事件都需要开发者通过监听这三个事件来自行处理。

2. 体感控制的设计与实现

iPad2和iPhone4设备中配备了六轴陀螺仪，这为游戏的玩法提供了更大的空间，目前陀螺仪已经在赛车、射击类等游戏中广泛使用。

六轴陀螺仪，其实就是三轴加速度感应器和三轴角速度感应器。加速度感应器用来感知iOS设备当前的位置状态（相对重力的方向，倾斜度等）。角速度感应器用来感知iOS设备当前的运动状态，某一轴上的运动情况。在本课题的开发中，只使用加速度感应器。

六轴陀螺仪中的加速度感应器并不是以周遭环境为参考系的，而是以设备本身为参考系的，根据直角坐标系设立x、y、z轴。x轴平行于iOS设备腰部，y轴垂直于iOS设备腰部，z轴则垂直于iOS设备屏幕。

加速度感应器就是监控重力加速度在x、y、z三轴上的加速度分量。如iOS设备竖

向竖直放置时，ax为0，ay为-9.8，az为0；iOS横向竖直放置时，ax为9.8，ay为0，az为0；iOS水平放置时，ax为0，ay为0，az为9.8。

要使用加速度传感器，就是获取重力加速度在iOS设备上的加速度分量。获取ax、ay、az的方式如下：

```
window.ondevicemotion = function(event) {
        ax = event.accelerationIncludingGravity.x;
        ay = event.accelerationIncludingGravity.y;
        az = event.accelerationIncludingGravity.z;
    }
```

监控这三个值的变化就可以随时获得iOS设备当前摆放的物理状态，通过这些物理状态来控制游戏。

在本游戏中，只需要iOS设备像汽车方向盘一样左右转动来控制player的左转和右转。这时，只需要监控ax的变化，iOS设备竖向垂直时，ax为0。当设备左转是，ax逐渐减小，当设备右转时，ax逐渐增大。只需设立一个临界值mixdir[来源]，当ax大于这个值时调用右转函数，当ax小于mixdir的负值时，调用左转函数，当ax介于二者之间时，调用速度保持函数，这样就达到了体感控制player转向的目的。体感控制监控更新函数如下：此处mixdir为1。

```
function dirStatusUpadte() {
    if ( ax < -1 ) {
        if ( dir != -1 ) {
            dir = -1;
            wsend("ctlDirLeft")
        }
    }
    else if ( ax > 1 ) {
        if ( dir != 1 ) {
            dir = 1;
            wsend("ctlDirRight")
        }
    }
    else {
        if (dir != 0) {
            dir = 0;
            wsend("ctlDirNone");
        }
```

```
    }
}
```

6. 目前存在的问题

自我评价，项目的不足存在两个方面：

3. 浏览器本身的问题导致的不足

这是指由于HTML5尚处草案阶段，浏览器对HTML5的实现良莠不齐。其中Web Socket还没有被大部分浏览器实现，而且对于Canvas的支持，各大浏览器并未实现像对CSS3一样的硬件加速支持，所以目前Canvas相对Flash，性能方面仍有距离（Flash支持硬件加速）。

对于这个问题，唯一的解决方法就是静待浏览器厂商的支持。根据目前HTML5的发展情况以及Google、Apple、Microsoft等厂商对其的支持状况，相信这一天并不会太远。

2. 项目实现过程中的不足

项目实现的不足之一是未实现玩家账号的管理。设计之初，由于游戏主体处理过程处于客户端（浏览器）内，所以简化了服务器端对用户的认证和处理过程。

与之对应服务器端加入账号管理并不困难。rails框架基于ruby语言，可方便的构建Web服务来管理用户账户。而项目架构设计上，已经将Web Socket 的连接独立出来，加入账户验证并与rails服务器整合也很方便[2]。开源项目已经建立，后续将逐步完善这部分内容。

项目实现的不足之二是游戏设计不够完整，可玩性不强。这也主要在于项目立题与对新技术的探索，没有强调游戏可玩性设计。这一点暂时不作调整。

7. 结论

目前商业游戏领域是否应该大规模使用HTML5技术？

不适合。分析如下：

(1) 整体来说HTML5上存在兼容性问题，特别是国内IE6占绝大部分浏览器市场份额的状况；

(2) Flash已积累大量的开发类库及框架，还有Flash CS、Flash Builder等成熟的开发工具，这方面HTML5尚有较大距离；

(3) 需要看到的是HTML5是业界公认的标准，发展速度迅速，以上问题可能在很短

的时间（1-3年）内得到解决。

目前那些专有领域适合使用HTML5做开发?

(1) 移动终端（iOS、Android、WebOS等移动设备）对HTML5的支持较好，制作Web App应用或游戏已经可行;

(2) Chrome系浏览器（ChromeBook笔记本和Chrome浏览器）对HTML5支持很好，已经可以使用大部分HTML5技术制作相关应用或插件;

(3) 聊天类、股票类对实时性信息要求较高的Web应用可以考虑针对高端浏览器用户使用HTML5技术提升其用户体验。

谢 辞

时光荏苒，大学四年匆匆而过。以此论文作为毕业的最后献礼，有无奈有遗憾也有不舍。

大学四年，不在乎专业的焚膏继晷，也不在乎爱情的轰轰烈烈，甚至不在乎桂林的好山好水，惟一在乎的是四年一起走过那些人。无论以后是否有机会再相见，甚至空留一句“君子之交淡如水“而后相忘于江湖。但这四年里的老师、同学、朋友们，便是这四年的全部。

“花有重开日，人无再少年”，纵然岁月如梭沧海桑田，时光河流的泥沙永无法掩盖这四年如宝石般青春的熠熠光彩。

感谢开源社区提供丰富的开发资源，让我的毕业设计拥有这些好玩的玩具。

感谢赵莹莹老师在繁忙的教学任务中指导并督促贪玩又拖沓的我完成毕业设计的相关内容。

感谢我自己，茫茫然亦步亦趋竟也走到今日。“乘骐骥以驰骋兮，来吾道夫先路！”

参考文献

1. **Dave Thomas, Chad Fowler, Angy Hunt et al.** *Programming Ruby* 中文版 第*2*版. [M]电子工业出版社, 2007. 215-222.
2. **Sam Ruby, Dave Thomas, David Heinemeier Hansson et al.** *Web*开发敏捷之道 第*3*版. [M]电子工业出版社, 2010. 113-128.
2. http://caniuse.com/
3. http://www.google.com/
4. http://html5test.com/
5. http://www.cnblogs.com/sanshi/archive/2009/07/14/1523523.html
6. https://github.com/dansimpson/em-websocket-server
7. http://docs.jquery.com/Main_Page
8. http://redbug0314.blogspot.com/search/label/Crowd-Simulation
9. http://zh.wikipedia.org/wiki/HTML5
10. http://stackoverflow.com/questions/4767268/best-ruby-on-rails-websocket-tool

附 录

附录1：项目中所使用的第三方库及简介

(1) jQuery http://jquery.com/

*“jQuery*是一套跨浏览器的*JavaScript*库，强化*HTML*与*JavaScript*之间的操作。由*John Resig*在*2006*年*1*月的*BarCamp NYC*上释出第一个版本。目前全球有*28%*的网站使用*jQuery*，是目前最受欢迎的*JavaScript*库*[1][2]*。

*jQuery*免费且为开放源代码，使用*GPL*和*MIT*许可证双协议*[3]*。*jQuery*的语法设计使得许多操作变容易，如操作文档对象（*document*）、选择*DOM*元素、动画效果、事件处理、发展*Ajax*以及其他功能。除此之外，*jQuery*提供*API*让开发者将自己所写的功能融入*jQuery*内。*”*

以上内容摘自维基百科：http://zh.wikipedia.org/wiki/Jquery

(2) Simple JavaScript Inheritance http://ejohn.org/blog/simple-javascript-inheritance/

jQuery 创始人John Resig 关于javascript 继承机制的一个实现，调用方式优雅、结构清晰。代码分析可参见：http://www.cnblogs.com/sanshi/archive/2009/07/14/1523523.html

(3) eventmachine http://rubyeventmachine.com/

官网介绍：fast, simple event-processing library for Ruby programs.

(4) em-Websocket http://rubyGems.org/Gems/em-Websocket

基于eventmachine对Web Socket构建的ruby语言开源实现。gitbug上有相应server、client实现DEMO的开源项目：https://github.com/igrigorik/em-Websocket

附录2：柏兹模型相关

*“*柏兹模型*(Boids Model)*

*“1987*年*9*月，蘭頓在羅沙拉摩斯籌備的第一屆國際人工生命研討會，得到了廣泛的迴響，共一百五十多名來自世界各地從事相關研究的學者與記者參加。各國學者在會議中都興致勃勃且迫不及待地展現自己獨自摸索多年的寶貝，有人談模擬演化的計畫、有人談胚胎發育的模擬演化、也有人談機器人，不過都沒有雷諾茲*(Craig Reynolds)*的柏茲模型讓人印象深刻。*”*

……

“雷諾茲受邀於蘭頓，在人工生命研討會上展示他的柏茲模型。柏茲模型中的每隻柏茲*(boid)*，皆具有一個有限的視野範圍並賦與自主運動的能力，當其它柏茲進入自己的有限視野範圍時，便遵循三個簡單的區域性規則*(local rule)*與其它的柏茲互動*[8]*，如圖七:

“避免碰撞*(Collision Avoidance):* 與鄰近的柏茲保持一個最小距離，以避免碰撞。

“對齊速度*(Velocity Matching):* 和鄰近柏茲的平均速度盡量保持一致。

“向中心凝聚*(Flock Centering):* 向鄰近柏茲的平均位置移動。 ”

……

“電腦模擬開始時，雷諾茲讓柏茲全完隨機散布在電腦螢幕上。緊接著，兩兩相鄰的柏茲便開始一同飛行，並沿路聚集其它的柏茲，自然而然地形成群體飛行的姿態。即使環境中有障礙物，他們也會自動地暫時分開以避開障礙物，並在閃過障礙物後會合、如圖八所示。偶爾柏茲難免撞上障礙物，在原地轉了幾圈而失去方向，不過一旦其它柏茲從旁飛過，它便會立刻重新加入飛行的隊伍，一切看起來都相當地自然。”

以上内容摘自：混沌邊緣的藝術- 人工生命 http://redbug0314.blogspot.com/2009/06/blog-post.html