Report for 3D Runner Game in VR

一般包括用了什么技术、框架，主要工程模块的功能描述，达到的最终结果的简单表述等等。

基本介绍

本项目是一个基于Web3D的跑酷类VR游戏。在游戏中，玩家可以操纵小球进行左右移动，并躲避路上的障碍。由于采用了3D框架，玩家可以用鼠标调整视角。在实现过程中，主要采用了a-frame框架来完成3D场景的搭建。

框架介绍

A-Frame 是一个用来构建虚拟现实（VR）应用的网页开发框架。由WebVR的发起人Mozilla VR 团队所开发，是当下用来开发WebVR内容主流技术方案。WebVR是一个完全开源的项目，已成长为领先的VR社区。

A-Frame基于HTML，容易上手。但是A-Frame不仅仅是一个3D场景渲染引擎或者一个标记语言。其核心思想是基于Three.js来提供一个声明式、可扩展以及组件化的编程结构。

A-Frame支持主流VR头显如Vive, Rift, Daydream, GearVR,Cardboard, 甚至可被用于增强现实（AR）。虽然A-Frame支持全谱，A-Frame的目标是定义具有位置跟踪和操控的完全身临其境和交互式VR体验，超出基本的360° 内容呈现。

功能模块

- 3D场景搭建

  本场景主要构成组件有天空、海洋、浮冰、冰面、障碍物与小球。由于海洋结构较为复杂，我们采用了内置开源代码的方式来实现海洋的结构与漂浮运动。除此之外，其余结构都由a-frame框架构成。

  对于绝大多多数场景，我们都通过标签添加上了动态运动与光影效果。部分实现代码如下所示：

  光源：

```
<!-- Lights! -->
<a-entity light="
    type: directional;
    castShadow: true;
    intensity: 0.4;
    color: #D0EAF9;"
  position="5 3 1"></a-entity>
<a-light intensity="0.8" type="ambient" position="1 1 1" color="#B4C5EC"></a-light>
```

  浮冰：

```
<!-- Ice! -->
<lp-cone class="iceberg" segments-radial="5" segments-height="3" height="1" amplitude-variance="0.25"
  radius-top="0.15" radius-bottom="0.5" position="3 -0.1 -1.5">
  <a-animation attribute="rotation" from="-5 0 0" to="5 0 0" repeat="indefinite" direction="alternate">
  </a-animation>
  <a-animation attribute="position" from="3 -0.2 -1.5" to="4 -0.2 -2.5" repeat="indefinite" direction="alternate"
    dur="12000" easing="linear"></a-animation>
</lp-cone>
<lp-cone class="iceberg" segments-radial="7" segments-height="3" height="0.5" amplitude="0.12" radius-top="0.25"
  radius-bottom="0.35" position="-3 -0.1 -0.5">
  <a-animation attribute="rotation" from="0 0 -5" to="5 0 0" repeat="indefinite" direction="alternate" dur="1500">
  </a-animation>
  <a-animation attribute="position" from="-4 -0.2 -0.5" to="-2 -0.2 -0.5" repeat="indefinite" direction="alternate"
    dur="15000" easing="linear"></a-animation>
</lp-cone>
<lp-cone class="iceberg" segments-radial="6" segments-height="2" height="0.5" amplitude="0.1" radius-top="0.25"
  radius-bottom="0.25" position="-5 -0.2 -3.5">
  <a-animation attribute="rotation" from="5 0 -5" to="5 0 0" repeat="indefinite" direction="alternate" dur="800">
  </a-animation>
  <a-animation attribute="position" from="-3 -0.2 -3.5" to="-5 -0.2 -5.5" repeat="indefinite" direction="alternate"
    dur="15000" easing="linear"></a-animation>
</lp-cone>
```

  障碍物：

```
<a-entity data-tree-position-index="1" id="template-tree-center" class="tree tree-center" shadow
  scale="0.3 0.3 0.3" position="0 0.6 0">
  <a-entity mixin="foliage"></a-entity>
  <a-entity mixin="trunk" position="0 -0.5 0"></a-entity>
  <a-animation attribute="position" ease="linear" from="0 0.6 -7" to="0 0.6 1.5" dur="5000"></a-animation>
</a-entity>

<a-entity data-tree-position-index="0" id="template-tree-left" class="tree tree-left" shadow scale="0.3 0.3 0.3"
  position="-0.5 0.55 0">
  <a-entity mixin="foliage"></a-entity>
  <a-entity mixin="trunk" position="0 -0.5 0"></a-entity>
  <a-animation attribute="position" ease="linear" from="-0.5 0.55 -7" to="-0.5 0.55 1.5" dur="5000">
  </a-animation>
</a-entity>

<a-entity data-tree-position-index="2" id="template-tree-right" class="tree" shadow scale="0.3 0.3 0.3"
  position="0.5 0.55 0">
  <a-entity mixin="foliage"></a-entity>
  <a-entity mixin="trunk" position="0 -0.5 0"></a-entity>
  <a-animation attribute="position" ease="linear" from="0.5 0.55 -7" to="0.5 0.55 1.5" dur="5000"></a-animation>
</a-entity>
```

- 小球控制模块

  本模块负责控制小球的运动。我们可以通过a、d按键或者左右按键来控制小球的左右移动来躲避障碍物。同时，我们实现了在该游戏在手机端的适配。即通过手机旋转角度来判断左右移动。

```
function setupControls() {
  return mobileCheck() ? setupMobileControls() : setupDesktopControls();
}

function setupDesktopControls() {
  window.onkeydown = function (e) {
    startGame();
    switch (e.keyCode) {
      case 37:  // left
      case 65:  // a
        movePlayerTo(player_position_index - 1)
        break;
      case 39:  // right
      case 68:  // d
        movePlayerTo(player_position_index + 1)
        break;
      default:
        break;
    }
  }
}

function setupMobileControls() {
  AFRAME.registerComponent('lane-controls', {
    tick: function (time, timeDelta) {
      var rotation = this.el.object3D.rotation;

      if (rotation.y > 0.1) movePlayerTo(0);
      else if (rotation.y < -0.1) movePlayerTo(2);
      else movePlayerTo(1);
    }
  })
}
```

```
function movePlayerTo(position_index) {
  if (position_index < 0) position_index = 0;
  if (position_index > 2) position_index = 2;
  player_position_index = position_index;

  position = { x: 0, y: 0, z: 0 }
  if (player_position_index == 0) position.x = POSITION_X_LEFT;
  else if (player_position_index == 1) position.x = POSITION_X_CENTER;
  else position.x = POSITION_X_RIGHT;
  document.getElementById('player').setAttribute('position', position);
}
```

- 障碍物生成模块

  我们通过设定三条赛道，并在赛道上随机生成障碍物的方式来完成地图的设计。这种设计方案可以确保每次运行都能生成不同的地图，提高可玩性。同时，设定障碍物由远及近运动，将玩家与摄像机固定，可以构造出玩家在向前运动的视觉效果。这样的方案可以减少计算量。

```
function addTreesRandomly(
  {
    probTreeLeft = 0.5,
    probTreeCenter = 0.5,
    probTreeRight = 0.5,
    maxNumberTrees = 2
  } = {}) {

  var trees = [
    { probability: probTreeLeft, position_index: 0 },
    { probability: probTreeCenter, position_index: 1 },
    { probability: probTreeRight, position_index: 2 },
  ]
  shuffle(trees);

  var numberOfTreesAdded = 0;
  var position_indices = [];
  trees.forEach(function (tree) {
    if (Math.random() < tree.probability && numberOfTreesAdded < maxNumberTrees) {
      addTreeTo(tree.position_index);
      numberOfTreesAdded += 1;

      position_indices.push(tree.position_index);
    }
  });

  if (mobileCheck()) {
    mirrorVR.notify('addTrees', position_indices);
  }
  return numberOfTreesAdded;
}

function addTreesRandomlyLoop({ intervalLength = 500 } = {}) {
  treeTimer = setInterval(addTreesRandomly, intervalLength);
}
```

- 碰撞检测模块

  当小球与障碍物碰撞时即游戏结束。我们通过检测如下条件：`POSITION_Z_LINE_START < position.z && position.z < POSITION_Z_LINE_END && tree_position_index == player_position_index`来判断小球是否撞上了障碍物。即小球的z轴位置在和障碍物的z轴体积重合，且小球在该障碍物的赛道内，即可完成碰撞判断。相比传统的碰撞检测，这种方式的计算量更小，通过先验知识来降低了程序的运算量，提高响应速度。代码如下：

```
const POSITION_Z_OUT_OF_SIGHT = 1;
const POSITION_Z_LINE_START = 0.6;
const POSITION_Z_LINE_END = 0.7;

AFRAME.registerComponent('player', {
  tick: function () {
    document.querySelectorAll('.tree').forEach(function (tree) {
      position = tree.getAttribute('position');
      tree_position_index = tree.getAttribute('data-tree-position-index');
      tree_id = tree.getAttribute('id');

      if (position.z > POSITION_Z_OUT_OF_SIGHT) {
        removeTree(tree);
      }

      if (!isGameRunning) return;

      if (POSITION_Z_LINE_START < position.z && position.z < POSITION_Z_LINE_END
        && tree_position_index == player_position_index) {
        gameOver();
      }

      if (position.z > POSITION_Z_LINE_END && !countedTrees.has(tree_id)) {
        addScoreForTree(tree_id);
        updateScoreDisplay();
      }
    })
  }
})
```

- 记分板模块

  每越过一个障碍物，记分板就会加一分。当物体撞上障碍物后记分结束，显示Game Over

```
var score;
var countedTrees;
var gameOverScoreDisplay;
var scoreDisplay;

function setupScore() {
  score = 0;
  countedTrees = new Set();
  scoreDisplay = document.getElementById('score');
  gameOverScoreDisplay = document.getElementById('game-score');
}

function teardownScore() {
  scoreDisplay.setAttribute('value', '');
  gameOverScoreDisplay.setAttribute('value', score);
}

function addScoreForTree(tree_id) {
  score += 1;
  countedTrees.add(tree_id);
}

function updateScoreDisplay() {
  scoreDisplay.setAttribute('value', score);
  if (mobileCheck()) {
    mirrorVR.notify('score', score);
  }
}
```