Baidu Smart Game-design, implementation and AI framework

Ping Wu, Architect of Baidu APP

W3C workshop on Web Game

2019
Agenda

1. Introduction
2. Runtime Design and Implementation
3. Developer Support
4. AI framework for the Smart Game
5. Future work
Introduction

1. **Baidu** - the leading Chinese language Internet search provider

2. **Baidu APP** – search and feed twin-engine powered mobile App, more than 180 million daily active users

3. **Baidu Smart Mini Program** – to take advantage of an emerging trend in China where apps with lower frequency are connecting to super apps to bypass the ever-rising cost of app pre-installs
Goals, Definitions and Challenges for Baidu Smart Game

Goals

1. Increase consumable resource type for Baidu App, set up game scenario to improve increase in DAU, user time, stickiness
2. Set up mini game ecosystem, explore distribution and monetization patterns

Definitions

1. Smart game is runnable in a browser-like environment built-up by super app, which is instant play without installation
2. A super container for forms like web game, smart mini game and cloud game
3. Include light, medium and serious game types

Technical challenges for Smart Game

- **Foundation layer**
  - Runtime architecture and implementation

- **Experience layer**
  - Core user experience-high performance, low latency, high-availability

- **Ecosystem layer**
  - Easy-to-develop, scalable game importing and rich distributable game types
Smart Game runtime design

Runtime Kernel:
- Construct a pure JS execution and native rendering environment with canvas 2D and 3D implementation.
- Provide an app model abstract for web game, like a web app package, lifecycle, version control, multi-app management.
- Provide rich native APIs in Baidu APP by JS binding framework.
JSB binding Framework - general and efficient JS-Native operation

Design principal:
• efficient
• general

Key elements:
• object model
• Life cycle
• Memory management
JSB binding framework - 10-200x improvement compared with JS bridge scheme

Loop 1000 for a simple function

```javascript
// Javascript impl
function testTimeFunc() {
  var x = 0;
  x = x + 2 + x * 6;
}

swanNative.console.time();
for (var i = 0; i < 1000; i++) {
  testTimeFunc();
}
swanNative.console.timeEnd();
```

<table>
<thead>
<tr>
<th></th>
<th>JS</th>
<th>JSB JAVA/OC implementation</th>
<th>JSB C++/C implementation</th>
<th>JS bridge with scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td>iOS</td>
<td>1.75 ms</td>
<td>0.65 ms</td>
<td>0.197 ms</td>
<td>2.15 ms</td>
</tr>
<tr>
<td>android</td>
<td>1ms</td>
<td>0.82ms</td>
<td>0.05ms</td>
<td>NA</td>
</tr>
</tbody>
</table>

Local data passing 1MB

```javascript
function request()
{
  swan.request({
    url: 'http://xxxxxxx',
    method: 'GET',
    responseType: 'text',
    dataType: 'json',
    success: res => {
      
    }
  });
}
```

<table>
<thead>
<tr>
<th></th>
<th>JSB text</th>
<th>JSB ArrayBuffer</th>
<th>Scheme with text</th>
<th>Scheme with ArrayBuffer</th>
</tr>
</thead>
<tbody>
<tr>
<td>iOS</td>
<td>11.5ms</td>
<td>3.5ms</td>
<td>795ms</td>
<td>1172ms</td>
</tr>
</tbody>
</table>
Efficient JS execution engine

- V8
  - Minimize parse and compile time
  - Code Cache
  - GC Incremental marking
  - GC at each game over

initialization stage optimization by v8
Design key elements:
- A lightweight standard implementation of HTML5 canvas 2D and 3D features in WebGL 1.0/openGL 2.0
- Fonts render with FreeType/CoreText
- Image decoder

Key problems:
- **Context Switch**
  - Minimize context switch frequency and must-to-update switching states by diff
- **texSubImage(canvas,..)**
  - Optimize CPU data copy with CopyTexImage2D
- **Font Render**
  - Arrange memory areas based on font size
Threading model design in native rendering environment

- **Main Thread** for JS execution and rendering engine.
- **Parallel image decoder**, image IO, audio IO from render thread.

**Design key points:**
- main thread for JS execution and rendering engine.
- Parallel image decoder, image IO, audio IO from render thread.
Key modules for Smart Game – file system, audio and network

- **Audio**
  - Optimize cache implementation
  - JSBinding
- **File system**
  - Optimize file IO
  - JSBinding
- **Network**
  - multi-threaded download
  - Background download

sprint stage audio optimization
Benchmark for Smart Game Runtime

Micro benchmark cases
- Image
- Render
- FS

Real world cases
- Laya
- Aquarium
- Box2D

UI Framework
- PIXI.js
- Gown.js

Benchmark Framework
- Cases Manager
- Cases Runner
- Performance Manager

Baidu Smart Game Runtime
- Benchmark Glue
- Image
- File System
- JSEngine
- Render
interface _naSwan.Performance {
    double now();
    // 返回当前的 CPU 占用
    float getCPU();
    // 返回当前的进程内存
    int getMemory();

    // 注册 GC 回调函数
    void registerGCCallback(Function);
    /* 以下接口目前 android 独有 */
    // 详细的内存信息
    MemoryInfo getMemoryInfo();
    // 设置、进程、线程CPU时间变化量
    CpuUsageTime getCpuUsageTime();
    // V8虚拟机的堆内存信息
    HeapStatistics getV8HeapStatistics();
    HeapSpaceStatistics getV8HeapSpaceStatistics();
}
Smart Game Benchmark

Score: -
系统: ios(iOS 12.2)
机型: iPhone(iPhone 8 <iPhone10,1>)
版本: 11.7.0.1
场景测试
WebGL Aquarium
Laya 3D 场景测试
接口单项测试
Image
File System
Network
Render

Score: 3324
系统: ios(iOS 12.2)
机型: iPhone(iPhone 8 <iPhone10,1>)
版本: 11.7.0.1
平均帧率: 59
加载时间: 1151 ms
运行帧数: 887
运行时间: 15008 ms
接口单项测试
Image
File System
Network
Render

Baidu APP scan code
Developer Support for Productivity

- 60,000 smart program developers
- 150,000 smart programs created
- 20,000 smart programs online
- 100 million DAU
Scalable large-scale game import with main stream game engines


- Laya

- Unity
Super container for Cloud Game, Smart Game and Native Game

Cloud game: based on video stream or gl commands stream

Veloce: Instant APP invocation

WASM: Web-Assembly version

Smart Game: based on pure JS execution environment and native rendering.
AI Framework for Baidu Smart Game
Challenges for AI enabled Smart Program and Game

- Rich and easy-to-use AI capabilities
- Same user experience as native app
- Flexible and general programmable abstraction
- Customized AI capabilities and application
- Framework for vertical industry and user scene
# AI framework for Baidu Smart Program and Game

<table>
<thead>
<tr>
<th>Smart APP</th>
<th>电商</th>
<th>社区</th>
<th>政务</th>
<th>工具</th>
<th>出行</th>
<th>教育</th>
<th>金融</th>
<th>旅游</th>
<th>游戏</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AI enabled</strong></td>
<td><strong>Smart User Interaction</strong></td>
<td><strong>Smart Recommendation</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Ability Framework, component, API</strong></td>
<td>语音控制</td>
<td>语音播报</td>
<td>卡证识别</td>
<td>人脸检测</td>
<td>商品推荐</td>
<td>AR相机</td>
<td>人脸AR</td>
<td>视频超分</td>
<td>能力扩展Paddle.js</td>
</tr>
<tr>
<td></td>
<td>语音</td>
<td>视觉</td>
<td>自然语言</td>
<td>知识图谱</td>
<td>增强现实</td>
<td>情景感知</td>
<td>人眼Lens</td>
<td>视觉跟踪</td>
<td>视觉理解</td>
</tr>
<tr>
<td></td>
<td>语音识别</td>
<td>图像合成</td>
<td>词法分析</td>
<td>知识理解结构化数据</td>
<td>三维感知AI渲染</td>
<td>场景感知设备状态</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>语音合成</td>
<td>图像合成</td>
<td>文本审核</td>
<td>文本审核</td>
<td>视觉计算</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Foundation</strong></td>
<td><strong>小程序/小游戏Runtime</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Data, Algo, computing</strong></td>
<td><strong>PaddlePaddle</strong></td>
<td><strong>Paddle-Mobile</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>百度AI开放平台</td>
<td>云端数据、算法、算力</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>百度APP移动端 + 移动端数据、算法、算力</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Architecture for Smart Game AR and Face

Game engine (Laya3D Three.js Cocos)
- Camera
- Scene Graph
- Light
- Skeleton
- Point Cloud
- Mesh

Camera matrix
Anchor matrix
Ambient Intensity
Joint matrix
Points Data
Face Mesh Data

swan.requestXRSession({ mode })

XRSession
mode: ar | face

DuXRSessionFaceMode
mode: du_face

DuXRSession
mode: du_ar

Video
view matrix
projection matrix
skeleton
landmarks

Smart Game rendering framework
Algorithm Framework

GPU-GPU textures copy

Smart Game video

Algorithm

ARKit (iOS)
ARCore (Android)
Algo Tracking, Detection, Skeletal

Inference Engine (Paddle Mobile)
OpenCV
Boost
libyuv
Design & Implementation rules for AR and Face

• **JS Framework**: compatible with WebXR, ARCore, ARKit API definitions, abstract developer-friendly JS object like Session, Frame, Video

• **Game runtime**: Communicate efficiently between JS and algorithm using JSB binding framework. Render the camera texture and canvas texture in the same hybrid rendering layer.

• **Algorithm**: Utilize AR ability of the system ARKit and ARCore and Baidu DuMix AR platform and Face Algo as a supplement
1. init Session：swan.requestXRSession({ mode: ‘ar’ })
   异步接口，初始化成功进入 success 回调，失败进入 fail 回调

2. getFrameData：xrSession.getFrame()
   • getViewMatrix(), getProjectionMatrix()
     三维相机的变换矩阵，可以直接用于 THREE.js 等 3D 引擎
   • getPointCloud()
     特征点点云，包含点云的(x, y, z) 的三维坐标
   • getTrackablePlanes()
     获取检测到的空间中的平面
   • hitTest(x, y)
     根据屏幕像素坐标得到三维空间坐标
   • createAnchor(poseMatrix)
     创建锚点，锚点是三维空间中一个固定的位置和朝向
   • getLightEstimate()
     环境光强度

3. close Session：xrSession.end()
1. **Init Session**: `swan.requestXRSession({ mode: 'du_face' })`
   - `mode: 'du_face'` 初始化人脸 AR 模式，未来会支持更多的模式去支持更多的 AR 场景（人体骨骼、物体追踪、图像追踪）

2. **getFrameData**: `xrSession.getFrame()`
   - `getCameraVideo()`
     - 相机 Video 对象，可以直接用于 THREE.js 等引擎作为纹理数据
   - `getUpdatedTrackableFaces()`
     - 当前帧检测到的人脸
   - `face.poseMatrix`
     - 人脸姿态的 4x4 变换矩阵
   - `face.landmarks`
     - 人脸特征点归一后的 (x, y) 坐标
   - `face.skeleton`
     - 人脸骨骼节点的变换矩阵
   - `face.blendShapes`
     - 表情系数

3. **close Session**: `xrSession.end()`
cc.Class({
  onLoad() {
    swan.requestXRSession({
      mode: 'du_face',
      drawCameraBackground: false,
      success: xrSession => {
        this._cameraVideo = xrSession.getCameraVideo();
      }
    });
  },
  update() {
    this.cameraTexture.update({
      image: this._cameraVideo
    });
  }
});
const frame = this.xrSession.getFrame();
const face = frame.getUpdatedTrackableFaces()[0];
if (face) {
    const videoBlendShapes = face.blendShapes;

    const a = videoBlendShapes.map(bs => bs.weight);
    const b = refBlendShapes.map(bs => bs.weight);

    this.totalScore.string = (similarity(a, b) * 100).toFixed(0);
}
const frame = xrSession.getFrame();
const face = frame.getUpdatedTrackableFaces()[0];
if (face) {
    const skeleton = face.skeleton;
    updateMeshSkeletons(skeleton);
}
Face Game Demo
AR Demo

Black

AR Mode
WebML & Paddle.js - customize your own AI application

1. loadModel(url | id) => Model
2. Model.predict(Tensor) => Tensor
3. createTensorFromImage(Image) => Tensor
Future Work

1. Smart Game is a very promising web tech powered by native experience
2. Provide more building blocks, more AI enabling features, more prosperous game society
Thanks !