WOTD

51CTO

全球 软件开发技术峰会

World Of Tech 2017

2017年12月1日-2日 • 深圳中洲万豪酒店

DEVELOPMENT

WOTD

51CTO

ECS实体组件模型

在游戏开发中的应用

kevinan

腾讯互娱，高级工程师

注：请您在提供照片后，
我们为您进行修改

目录

个人履历

01 腾讯游戏，后台开发，高级工程师

02 参与或主导过的项目：

03 《QQ农场》、《疯狂联盟》、《火影忍者》

04 目前负责一款“多人FPS手游”

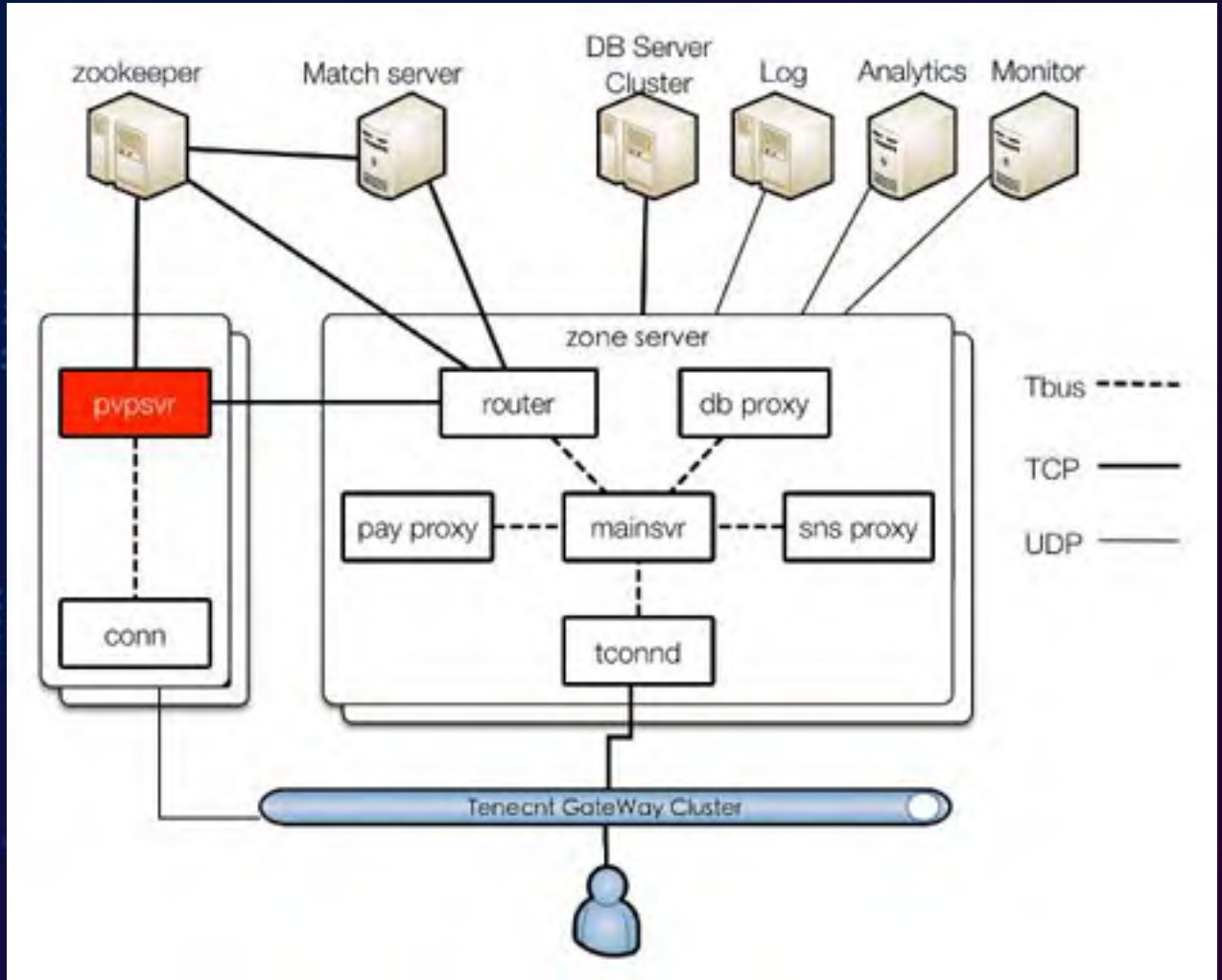
话题背景 – 游戏开发进化史

01 Web Game, 请求响应式

02 RPC，框架，数据分区

03 Scalability，容灾，协程

04 PVP & Authoritative Server


zookeeper
Match server
DB Server Cluster
Log
Analytics
Monitor
zone server
pvpsvr
router
db proxy
pay proxy
mainsvr
sns proxy
conn
tconnd
Tbus
TCP
UDP
Tenecnt GateWay Cluster

话题背景 – 权威服务器

01 后台逻辑开始变得复杂

02 反外挂、竞技向

03 状态同步

04 游戏逻辑全量模拟

话题背景 –游戏逻辑全量模拟

01 节奏快，网络延迟容忍度低

02 前后台开发融合

03 需要一个高效的编程模型

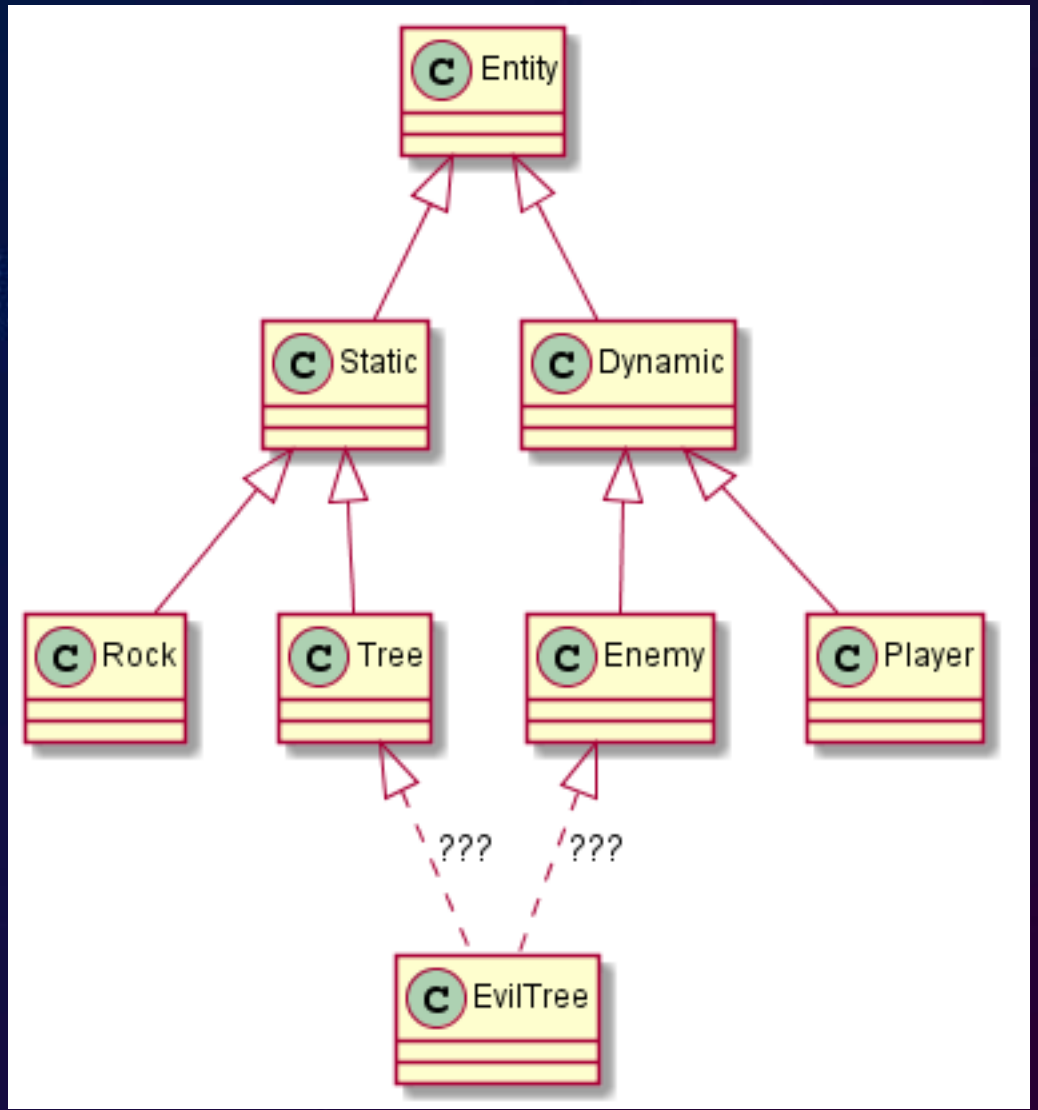
04 降低代码复杂度

游戏到底是如何做的呢？


消音器
手枪子弹步枪子弹
AK空投游泳枪托
冲锋
绷带瞄准镜头盔
卡丁车
M
爬山
Kar
LYB狙击手枪
玩家匍匐
伏地魔
三轮车
吉普车
防弹衣

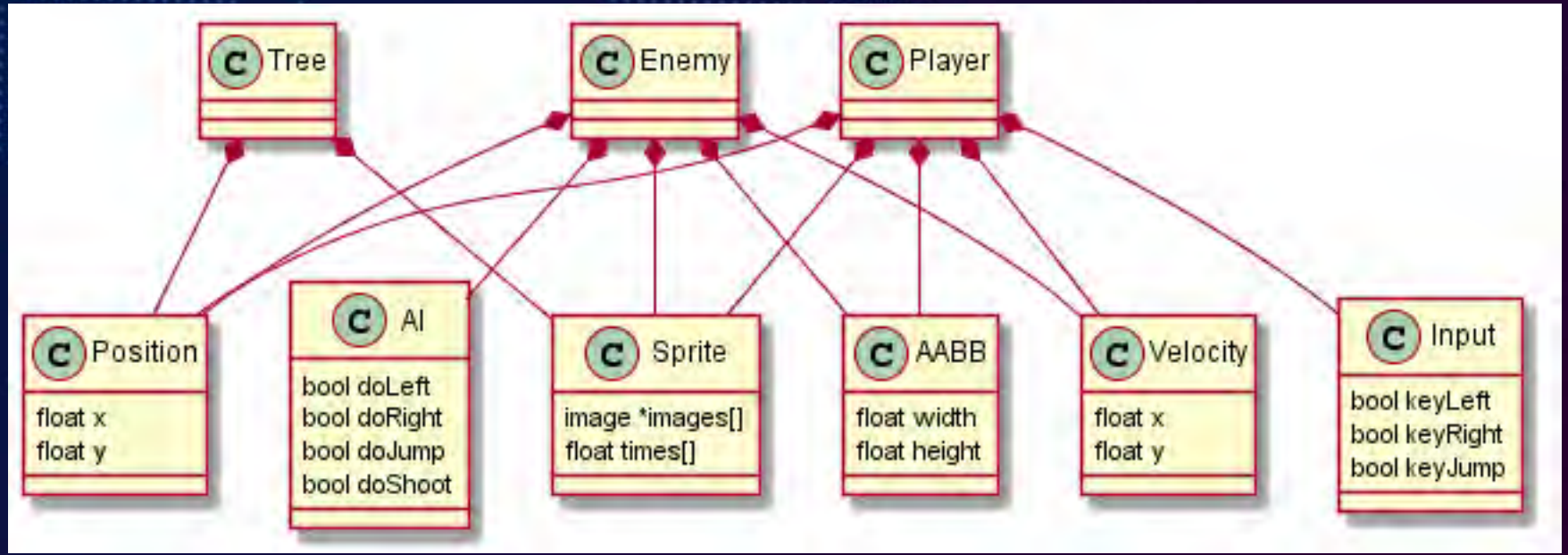
WOTD

OOP



51CTO

Entity & Component



Terminology

01 实体（Entity）

02 组件（Component）

Game Engine Tick

```
For (auto * entity : entities) {
        entity->Update();
}
```

```
For (auto * c : components) {
        c->Update();
        // Update里有大量逻辑
}
```

Entity Component vs. OOP

01 修改游戏逻辑，无需程序员参与

02 立项时无法确定所有实体的相互关系

03 常常有些策划需求是要“横切”传统OOP

04 编译更快，测试更快，调试更快

05 开发起来更加敏捷

EC实战

对象如何获得组件

01 如果这个类创建了自己的组件 

02 如果由外部代码提供组件 

组件之间如何传递信息

01 通过修改容器对象的状态
- 组件间解耦
- 浪费内存、顺序依赖

02 直接互相引用
- 简洁高效
- 紧耦合、hardcode

03 通过传递信息*

04 间接引用*

GetComponent<Movement>()

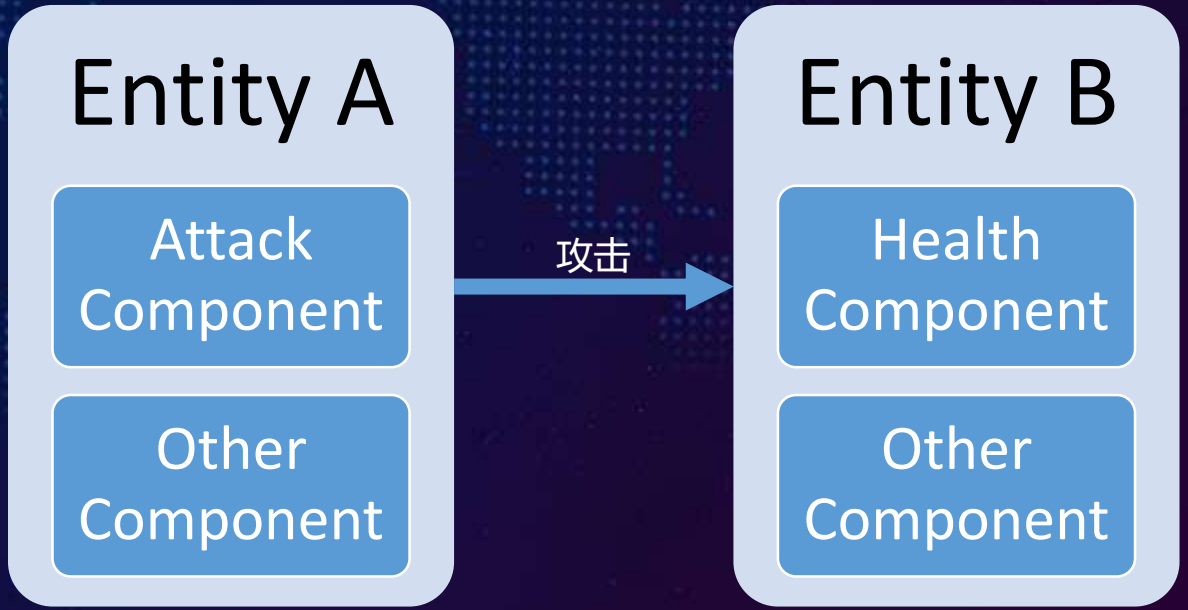
消息队列

01 中心事件队列 & Ringbuffer

02 优先级管理

03 事件回调 – C++11的std::bind 绑定“类成员函数”

痛点1：伤害计算的纠结？


Entity A
Attack Component
Other Component
攻击
Entity B
Health Component
Other Component

痛点2：网络同步失败时，回滚逻辑复杂

01 状态同步，需要Client做预测

02 预测失败时要回滚状态，解决冲突

03 Component中的逻辑不容易回滚

What is ECS

01 实体-组件-系统 (ECS)模式

02 是一种主要用于游戏软件开发的架构模式

03 组合优于继承原则，减少继承层次结构

04 运行时可以改变实体的行为（动态增删组件）

05 常与“面向数据”的设计技术相结合

Terminology

01 实体（Entity）

02 组件（Component）

03 系统（System）*

ID only

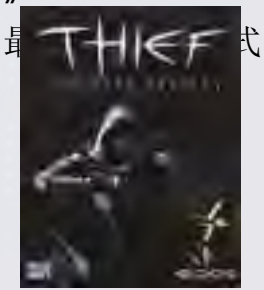
没有方法

没有字段

History of ECS

1998

- 《神偷:黑暗计划》
- 最[illegible]式



2002

- Scott Bilas
- 《地牢围攻》

2007

- Adam Martin
- 《闪点行动：龙之崛起》

2008

- Unity引擎
- 广泛引用

unity

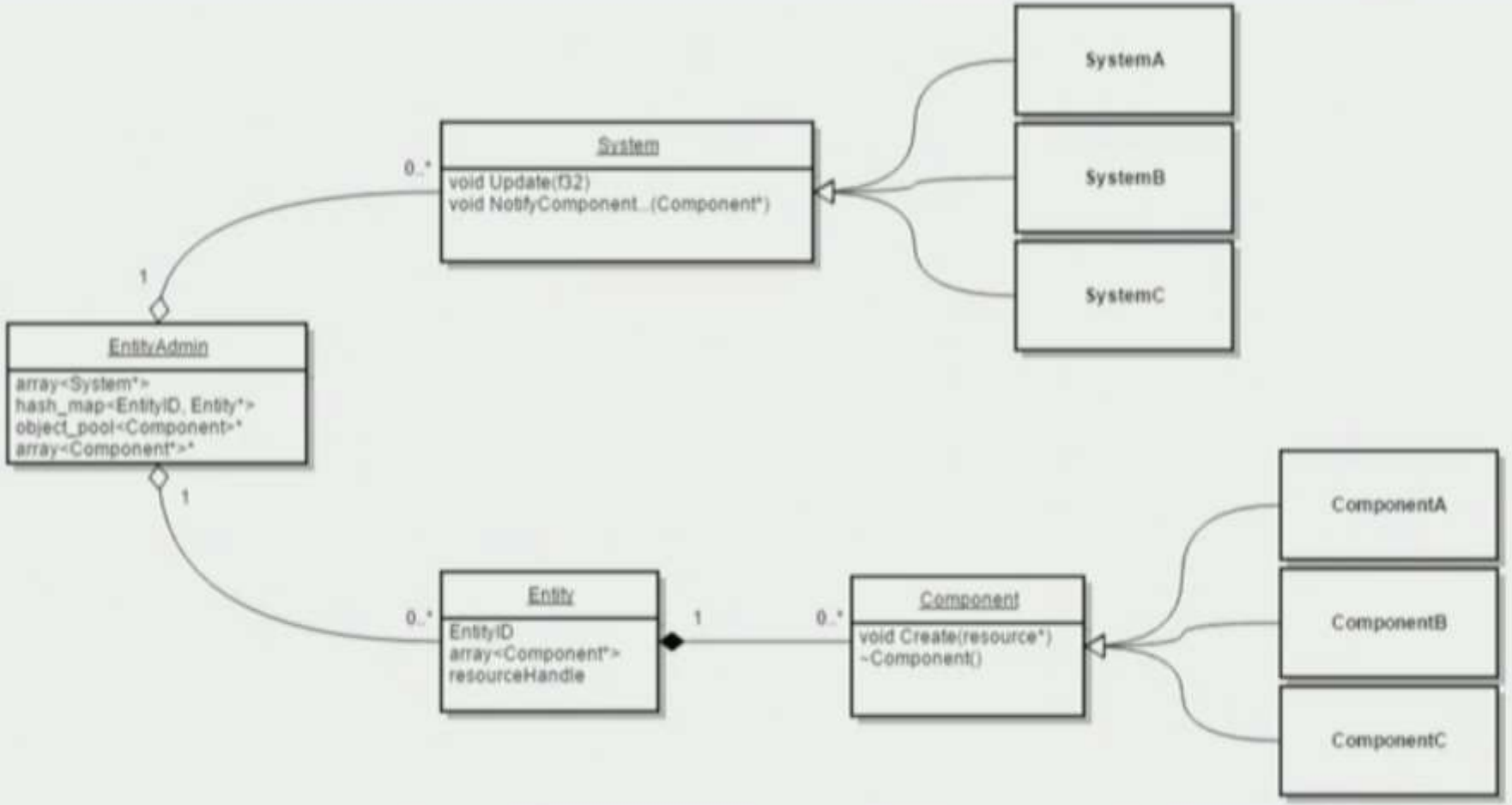
发扬光大

2016 暴雪

Overwatch《守望先锋》

MOBA + FPS

GDC2017翻译


System
void Update(f32)
void NotifyComponent...(Component*)
SystemA
SystemB
SystemC
0..*
1
EntityAdmin
array<System*>
hash_map<EntityID, Entity*>
object_pool<Component>*
array<Component*>*
1
0..*
Entity
EntityID
array<Component*>
resourceHandle
1
0..*
Component
void Create(resource*)
~Component()
ComponentA
ComponentB
ComponentC

ECS实战

Game Engine Tick - ECS

```
For (auto * sys: systems) {
        sys->Update();
}
```

```
For (auto * c : Components) {
        // do something
}
```

伤害计算不再纠结


Attack System
Entity A
Attack Component
Other Component
Entity B
Health Component
Other Component

数据局部性

通过合理组织数据，利用CPU的缓存机制来加快内存访问速度

01 放弃指针

Entity里直接保存连续的Component数组

02 冷/热分离

数据量大的组件，拆分为两部分：根据是否需要每帧访问

缺点及应对

01 Singleton 组件

某类 Component 可以也应该只有一个，例如存放键盘输入的 Component

02 Utilities Function

不同的System都需要查询两个 Entity 的敌对关系

结束语

OOP → EC → ECS

Thank you!