# VRTest： 虚拟现实场景自动测试的可扩展框架

## 摘要

虚拟现实（VR）是一种新兴技术，吸引了培训、教育、远程通信、游戏和导航等各种应用领域的兴趣。尽管 VR 软件项目的数量不断增长，但 VR 软件的质量保证技术却没有得到很好的研究。因此，VR 软件的验证主要依赖于纯手工测试。在本文中，我们提出了一种名为 VRTest 的新型测试框架，用于自动测试 VR 软件中的场景。具体而言，VRTest 从 VR 场景中提取信息，并控制用户摄像头探索场景，通过特定的测试策略与虚拟对象进行交互。VRTest 目前支持两种内置测试策略： VRMonkey 和 VRGreed 使用纯随机探索和贪婪算法来探索 VR 场景中的可交互对象。我们工具的视频可在 Youtube 上观看：https://www.youtube.com/watch?v=TARqTEaa7_

## 介绍

虚拟现实（VR）是一种新兴技术[10]，有许多不同的应用场景，如游戏、虚拟展览和参观、培训、教育、产品设计和远程通信等。最近的一份市场报告[3]估计，2019 年 VR 市场总值将达到 115.2 亿美元，其中包括 19 亿美元的 VR 软件市场[5]。此外，预计在接下来的五年中，市场价值将以每年 48.7% 的速度高速增长。COVID-19 病毒的流行进一步加速了 VR 技术的应用。2020 年，数以千计的应用程序被上传到 Google Play [2]、Apple Store [1] 和 Oculus Market [4]。这些应用程序的下载量已超过来自世界各地的 1.71 亿用户[9]。与所有其他类型的软件一样，虚拟现实软件也需要测试来验证和提高其质量。然而，与许多其他软件领域（如软件库、服务器软件、图形用户界面软件）已经开发并部分采用了许多自动和半自动测试技术不同，虚拟现实软件的自动测试技术还没有得到很好的研究。VR 软件通常由一个或多个 VR 场景组成。每个虚拟现实场景代表一个三维空间，虚拟对象（相当于图形用户界面软件中的图标/控件[22]）在其中被放置和移动。用户通常通过指针点击（即在短时间内保持摄像机视图中心的白点指向可交互对象）与虚拟对象进行交互，从而操作软件。在本文中，我们提出了一个自动测试框架 VRTest，用于测试 VR 软件中的 VR 场景，并内置了 VRMonkey 和 VRGreed 两种测试策略。其中，VRTest 会自动定期（处理对象移动）定位虚拟空间中的所有虚拟对象，并根据测试策略移动/旋转摄像头。需要注意的是，VRTest 需要同时跟踪可交互对象和不可交互对象，因为后者可能会阻挡摄像机的移动路线并遮挡可交互对象。在 VRTest 的基础上，VRMonkey 是一种纯粹的随机探索测试策略。

总之，本文有以下贡献。- 我们探索并总结了自动测试 VR 场景的主要挑战。- 我们开发了一个自动测试 VR 场景的框架，该框架可与不同的测试策略相结合。- 我们开发了基于纯随机探索（VRMonkey）和贪婪探索（VRGreed）的两种内置测试策略。- 我们准备了一个包含五个顶级 VR 软件项目的数据集，可在该领域的未来研究中重复使用。

## 框架

在本节中，我们将介绍我们的 VRTest 框架，该框架可自动探索 VR 软件中的虚拟场景。VRTest 提供了一套基本操作来控制和配置测试过程。图 1 是 VRTest 框架的概览。从图中我们可以看出，VRTest 框架由五个主要部分组成：用于获取 VR 场景信息（即运行时状态）的 VR 场景监视器、为可交互的虚拟对象提供状态报告的虚拟对象仪器、控制所有信息传输和探索速度的探索控制器、测试配置接口以及为测试技术提供场景信息反馈和三个基本操作（移动、旋转和触发）的测试技术接口。

VR 场景监控器组件提取 VR 场景中虚拟对象的信息。由于虚拟对象可以在运行时创建和销毁，也可以从一个位置移动到另一个位置，因此信息提取过程是周期性的。特别是，该组件会提取以下主要类型的信息，因为它们对各种测试技术都很有用。- 虚拟对象的边界框。监视器会提取所有带有渲染器（使其可见并阻挡视线）的虚拟对象的边框（包围对象的最小矩形立方体）。监视器会进一步将带有碰撞器的虚拟物体与其他物体区分开来，因为这些物体可能会阻碍用户（和摄像机）在其中移动。需要注意的是，边界框（见图 2）只是虚拟物体形状的近似值，目的是简化后续计算。此外，虽然透明渲染器可能不会遮

挡视线，但我们不会单独处理这些物体。我们认为这些都是适当的近似值，因为虚拟物体的形状和透明度千差万别，而且这种近似值在测试中不会造成误判，而只会减慢探索过程（例如，当一个物体被另一个物体遮挡，但实际上在当前摄像机位置和角度下是可见的）。- 虚拟物体的位置。监视器会提取所有虚拟物体的三维坐标，这些坐标包括虚拟对象也可以通过克隆另一个现有对象来创建。由于这些对象副本通常具有相同的行为和附带的相同脚本集，对它们触发事件可能会覆盖完全相同的对象行为和代码部分。因此，在它们身上多次触发事件的效率会很低，而监控器会将这些信息提供给测试策略，以便它们进行优化。

2.2 虚拟现实对象仪器 当 VRTest 试图触发虚拟对象上的指针点击事件时，存在很多不确定因素。例如，对象的形状可能不规则，因此将用户摄像头对准其位置可能无法触发事件。此外，某些呈现器可能是透明的，因此用户摄像头可能会意外触发其他事件。为了确保 VRTest 始终掌握关于哪些虚拟对象已被覆盖的最新准确信息，我们在 VRTest 中设计了一个仪器组件。具体来说，仪器器将为所有可交互的虚拟对象的每个注册事件添加一个状态变化报告器（一个额外的报告事件处理程序方法）。一旦相应的事件被触发，状态变化报告器就会向 VRTest 报告，这样 VRTest 就能随时知道哪些对象和事件已被触发。我们用来插入状态变化报告器的代码如下所示。

2.3 勘探控制器 勘探控制器是 VRTest 的核心组件，控制着整个测试过程。它从 VR 场景监控器和状态变化报告器接收更新信息，并将信息提供给它所采用的测试策略。值得注意的是，Unity 框架会定期调用 Update（每帧）和 FixedUpdate（每 0.02 秒）方法来刷新 VR 场景。VRTest 将其探索控制器嵌入到 FixedUpdate 方法中，使探索过程与 VR 场景的帧刷新同步。我们选择 FixedUpdate 而不是 Update，是因为前者有固定的调用间隙，不受运行时 fps（每秒帧数）的影响，因此能更好地控制物理运动。如果动作已经完成，VRTest 将首先从监视器中提取 VR 场景信息，调用仪器器来检测新发现的尚未检测的虚拟对象，然后要求测试技术决定下一步要执行的动作（即触发、旋转、移动或任意组合）。

2.4 测试配置 VRTest 允许开发人员根据需要测试的 VR 场景的要求和特征来配置 VRTest。下面列出了一些主要的配置项目。- 移动范围。移动范围定义了用户摄像头可移动到的位置范围，测试策略在计算两个位置之间的可行路线时需要考虑到这一点。根据 VR 软件的特性，移动范围可能会有很大不同。例如，飞行模拟软件应用程序可能允许用户摄像机在很大程度上在所有三个维度上移动，但驾驶/行走模拟软件应用程序可能只允许在 $X$ 和 $Y$ 轴上移动，而不允许任何 $Z$ 轴移动。其他一些软件可能有更固定的移动模式（如沿线或路线移动），甚至不允许任何移动（如一些射击游戏）。VRTest 框架允许开发人员手动指定移动范围。- 旋转范围。与限制用户摄像头位置的移动范围类似，旋转范围也限制用户摄像头的观察角度。这在不同的 VR 应用程序中更为一致。通常情况下，Y 轴旋转（头部向左和向右转动）两侧的角度可达 180 度，X 轴旋转（头部向上和向下转动）的角度可达 90 度。原因是这个旋转范围与人类头部的旋转范围大致相同。因此，在我们的框架中，我们也将旋转范围设置。

2.5 测试策略接口 VRTest 为任何测试技术的实现提供了一个接口。该接口包括三种方法： 旋转、移动和触发。不同的测试技术可以实现这三种方法，以不同的策略探索 VR 场景。VRTest 为指针点击事件提供了默认的触发器实现，但它可以很容易地扩展到新的事件类型和触发事件的不同方式。此外，VRTest 还为测试技术提供了获取 VR 场景信息（即从 VR 场景监控器、状态变化报告器和测试配置接口收集的所有信息）的接口，以备实施测试策略时之需。

# 评估

为了评估我们的框架和测试策略，我们在 UnityList1 上的五个顶级 VR 软件项目中应用了 VRTest 的两种测试策略。我们从 UnityList[8]中收集主题项目，因为它是最大的开源 VR 软件项目库。在 UnityList 中，我们按照特色得分排名，只考虑至少有一个虚拟对象、至少有一个事件触发器的项目。表 1 列出了五个评估对象项目的基本信息。在表 1 中，我们分别列出了源文件数量、代码行数和静态虚拟对象/预制件数量（动态虚拟对象通常是通过克隆静态虚拟对象/预制件创建的）。为了进行评估，我们使用了 Unity 2019.4.2f1 版和 Visual Studio 2017（用于编译 C# 源代码），并在配有英特尔酷睿 i7-6500U CPU、8GB 内存和英特尔 HD 520 图形卡的计算机上运行实验。我们设置的超时时间为 100 秒。我们通过可交互对象覆盖率（即在所有可交互对象中触发可交互对象的比例）来衡量测试的有效性。对于可交互对象覆盖率，我们将同一类型的对象视为一个。结果如表 2 所示。表中第 2 列和第 3 列分别列出了 VRMonkey 和 VRGreed 的覆盖率。从表中我们可以看到结果。

# 讨论

我们的测试框架和测试技术目前侧重于指针点击事件类型，因为这是最常见的支持事件类型。某些设备还支持其他一些事件类型，如抓取事件（允许用户用虚拟手抓取某些虚拟对象）和碰撞事件（当用户摄像头处于相同位置或接近现有虚拟对象时，允许用户推动或收集某些虚拟对象）。由于这些事件主要是基于接触的事件（即用户摄像头需要非常靠近虚拟对象才能触发事件），因此与指针点击相比，触发这些事件的复杂性较低，因为我们不需要考虑对象遮挡等情况。更复杂的情况是，虚拟对象必须按一定顺序进行交互才能产生结果。我们的三种测试技术都没有刻意处理这种交互顺序，因此能否触发结果可能在很大程度上取决于重复触发可交互虚拟对象上的事件，而与这些对象相关的方法仍未涵盖在内。今后，我们计划使用静态分析来识别事件处理程序之间的依赖关系。基于这些依赖关系，VRTest 将能以更恰当的顺序触发事件，从而暴露更多的软件行为。

# 相关工作

我们不知道在虚拟现实软件测试领域现有的努力，但有一些关于游戏测试的研究。无极[25]是一个基于进化算法和强化学习的游戏自动测试框架。它能探索游戏空间和分支，并通过阶段的传递取得进展。Testmig [19] 可在移动设备间迁移测试用例，但其方法无法直接扩展到 VR 设备。Zhao 等人[24]提出了一种通过学习玩家动作序列来增强游戏测试中的游戏策略的方法。Bergdahl 等人[11]提出了一种通过强化学习来增强现有手动编写的测试脚本的方法。Molina 等人[15]提出的 VRDepend 是一种自动提取虚拟对象间依赖关系的方法，可用于分析和测试 VR 软件。然而，上述方法都主要针对游戏战术，是为 2D 游戏设计的，因此当应用于 3D 软件时，它们仍然面临着摄像机灵活移动/旋转以及访问视线外和遮挡物体的挑战，而这正是本文的重点。

此外，还有一些关于虚拟现实软件和视频游戏软件的实证研究。Murphy-Hill 等人[16]对视频游戏开发人员进行了研究，以了解视频游戏开发中的挑战以及这些挑战与传统软件开发的不同之处。Washburn 等人[21]对失败的游戏项目进行了研究，以找出游戏开发中的主要陷阱。Lin等人[14]研究了steam平台的常见更新，以了解游戏更新的优先级。Rodriguez 和 Wang。[20] 对开源虚拟现实软件项目进行了实证研究，以了解其受欢迎程度和常见结构。Pascarella 等人[18]对开源视频游戏项目进行了研究，以了解其特点以及游戏与非游戏开发之间的区别。Zhang 等人[23]研究了检测移动增强现实应用程序中潜在隐私泄露的可能解决方案。Li 等人[13]研究了基于网络的扩展现实应用程序中的错误特征。Nusrat 等人[17]研究了真实世界 VR 项目版本历史中的性能优化，并总结了新的 VR 开发人员应避免的主要性能问题。

# 总结

VR 软件的使用场景越来越多，但其质量保证技术仍然落后。在本文中，我们提出了一种名为 VRTest 的新型框架，用于自动测试 VR 软件。VRTest 从 VR 场景中提取信息，控制用户摄像头探索场景并与虚拟对象交互。在 VRTest 的基础上，我们进一步开发了两种测试策略： VRMonkey 和 VRGreed 分别使用纯随机策略和贪婪算法。我们还在 UnityList 中的五个虚拟现实软件项目上对测试策略进行了初步评估。通过对 VRTest 进行扩展，使用 Move、Turn 和 Trigger 函数的新实现，可以轻松添加新的测试策略。今后，我们计划朝以下方向努力。首先，对于 VRTest 框架，我们计划对其进行扩展，以支持更多类型的事件，如抓取事件和碰撞事件。其次，我们计划扩展测试策略，以考虑 VR 场景中更多的全局信息，并计划通过使用基于人工智能或基于搜索的技术来进一步增强我们的策略，这些技术已在图形用户界面测试中被证明是获取全局优化路线的有效方法。第三，我们计划用更多的对象和非基于 Unity 的软件项目来评估我们的框架。第四，某些软件行为可能只有在事件按特定顺序触发时才会暴露，因此我们计划使用静态分析来识别事件处理程序之间的依赖关系。基础