Dev-C++和Godot*的比较

刘功泽

摘要

集成开发环境（简称 IDE）是一种将程序开发过程中所需的各种工具集成在一起的应用程序，它为程序员提供了一个全面的、一站式的开发环境。游戏引擎是一种特殊的IDE，为游戏开发者提供了一系列编写好的工具集合。兼具强大的开发功能与高度的可编辑性，它充当了交互式实时图像应用程序的核心组件。

本文从一个初学者的视角，聚焦于二者的两个典型：Devc++和Godot，比较这两者在IDE功能实现上的相似点和差异之处。Godot作为一款更加现代化的IDE，在功能上做出了更全面的支撑，其直观性更方便于编程初学者直接上手编写代码。相应的，Dev-C++配置了极其简洁的编写环境，有着较长的发布历史，对于软件操作没有很高的要求，同时具有更为丰富的学习资料可供参考。

1 引言

编程的初学者通常从一门主流语言入门，比如C++，而Dev-C++作为一款较为基础的C++IDE成为了常见选择。它的界面简洁明了，没有过多复杂的菜单和工具栏，有利于初学者快速熟悉和找到常用的功能按钮，如编译、运行、调试等，将重心更多的放在编程语言本身，简化了摸索开发工具的使用方法的过程。

游戏引擎作为IDE的一种，既与其他IDE具有相似的架构和操作模式，又有它独特的功能特性。面对早期游戏开发周期长、成本高的局面，有经验的开发者借鉴先前的游戏设计，极大提高了开发效率，推动了游戏引擎的诞生。它使游戏开发者不再需要从头开始编写代码，减少了重复劳动。在当下环境，Godot以完全免费且开源的特性，以及其出色的2D渲染性能，获得了众多独立开发者和小型团队的青睐。

*Godot，一款开源的游戏引擎，发布于2014年

本文将以Dev-C++和Godot为例，聚焦于它们的在程序设计方面的一些功能特性，希望能增进读者对两种工具的理解

2 Dev-C++和Godot的比较

Dev-C++是Windows环境下的一个轻量级C/C++集成开发环境。Godot作为一款开放时间较短游戏引擎，具有很多优秀的功能，拥有较高的信任度。我们将在下面介绍二者作为IDE在一些相似功能实现上的差异。

一款IDE的基础功能通常由以下几部分组成：代码编译器，编译器（编译器集成），调试器，项目管理工具，图形用户界面设计工具。在这些方面，两者具有很多共通之处，但各自独有的特点也十分明显。下文将针对影响初学者入门难度的几个因素，对IDE基础功能中具有代表性的几个方面展开探讨，比较Godot和Dev-C++在程序设计实现上的一些差异。

2.1 功能支撑

2.1.1 代码编译

Dev-C++使用了 AStyle（Artistic Style）的代码格式化工具，形成了它独特的代码编辑风格。Godot虽然支持多种程序设计语言（包括C++），但原生GDscript脚本语言自然是适配引擎本身的最佳选择。

C++具有头文件包含机制，意味着在源文件的开头部分会包含一系列必要的头文件。虽然这有助于代码复用和模块化，可以引入外部的函数声明、类型定义等内容，但也成为了初学者编程时的一大困扰（奇怪的bug又多了……）。而Godot以场景和节点为核心的架构，使GDscript摆脱了传统头文件包含的限制，即函数和类的声明与定义分离。同时，它在设计上更注重简洁性和动态性（类似Python），函数和类的定义可以更加灵活，不需要像C++那样严格的头文件包含来管理声明和定义。

代码编辑功能上，Dev-C++ 的代码提示和自动补全功能相对较弱。在多文件开发中，当涉及到跨文件的函数调用、变量引用等情况时，不能及时准确地提供相关的提示和补全，一定程度上会影响到开发效率。在处理较为复杂的代码结构时，开发者需要花费更多的时间去记忆和输入函数名、变量名等。同时，它缺乏对代码重构的有效支持，重命名变量、提取函数、移动代码块等常见的重构操作实现十分麻烦（需要手动），不利于代码的维护和演进。在这些方面，作为新一代IDE，Godot具有显著优势。

编辑器界面与布局上，Godot具有可视化的场景编辑器。为开发2D和3D游戏而生，它内置了按钮（Button）、标签（Label）、文本框（LineEdit）等强大的工具，能完美地实现图形化界面（GUI）的设计和完善。而Dev-C++使用了较为传统的多窗口布局界面，这种布局更符合初学者的编程需求，适合专注于代码编写和调试的开发场景。

2.1.2 编译器集成

编译器是IDE不可或缺的翻译程序，它将高级程序设计语言编写的源代码转换为计算机能够理解并执行的机器语言。Godot和Dev-C++均实现了编译器集成，这意味着编程者不需要手动配置编译环境，可以高效地使用本地编译器进行程序开发。

C++中贯穿了结构化编程的思想，这也意味着无条件跳转语句（goto）的使用是不被建议的，它可能会破坏代码的线性逻辑，降低代码的可读性，甚至可能在变量的状态和资源管理方面造成混乱和错误。同时，它也与面向对象编程相悖。

然而在godot中，有一种类似功能却被广泛的使用，它便是Godot的信号（Signal）机制。信号作为一种事件通知机制，使得一个对象在特定事件发生时能够与其他对象建立联系，使得接收信息的对象就可以调用其槽函数作出相应的反应。Godot对信号功能提供了丰富的支持，通过使用内置信号或者自定义信号，程序可以实现节点间的连接和参数传递等操作。信号是游戏中实现交互功能的关键工具，同时在场景管理中起到了不可代替的作用。与goto不同，信号建立了明确的发送者和接收者关系，具有清晰的通信路径；同时，它采用了基于事件驱动的模型，运行过程具有高度的逻辑有序性。因此，它在不干扰程序正常运行的前提下，增强了代码的独立性和扩展性。

2.2 软件操作

项目管理的质量直接影响的开发工作最终质量。Dev-C++支持多文件项目开发，但是其项目管理功能有限，在很多方面存在不足之处。

首先，它的复杂项目层次结构支持和文件组织不够灵活。它虽然可以创建文件夹来组织源文件和头文件，但是难以按照功能模块、层次结构等方式对大量文件进行有效的组织和管理，不能满足大规模项目开发中对项目结构清晰性和可维护性的要求。

其次，它对库依赖的处理不尽人意。在进行较为复杂的程序开发时，使用

```
using namespace std;
```

调用c++标准库可能会造成名称冲突，导致意想不到的错误。在处理外部库依赖时，Dev-C++ 的操作相对繁琐且不够直观，开发者需要手动配置库文件的路径和相关链接选项；并且，Dev-C++本身没有很好的库版本管理功能，当你的项目依赖于多个库并且这些库有不同的版本要求时，可能会出现兼容性问题，容易出现配置错误导致编译失败。

但是，作为C++学习者的新手村，Dev C++最突出的优点是它可以简单实现单文件开发。相信下面的操作初学者再熟悉不过：在菜单栏中，选择“文件（File）→ 新建（New）→ 源代码（Source File）”。它可以针对单文件无需建立项目进行编译或调试，极大地降低了学习难度

相比之下，Godot在项目管理方面的实现就复杂得多。它的基本结构由场景树-场景-节点构成。

节点（Node），作为Godot 中的最小构建单元，是能够执行特定的任务（如显示图像、播放声音、处理输入、进行物理模拟等）的对象或组件。游戏引擎强大功能的实现，自然离不开它提供了各种各样的内置节点。如Sprite节点用于显示二维图像，AudioStreamPlayer节点用于播放音频，CollisionShape2D节点用于定义二维物体的碰撞形状等。同时，节点具有属性和方法，可以通过脚本进行访问和修改，这实现了它的可扩展性。开发者可以通过继承现有的节点类型或创建自定义节点类型来扩展功能以满足游戏的特定需求。

场景（Scene），由一个或多个节点组成，可以看作是具有一组特定功能的完整模块，在游戏开发中扮演着核心的角色。开发者可以将游戏划分为多个不同场景，每个场景负责游戏的一个特定部分，不仅使得游戏的结构更加模块化，同时也有助于实现代码和资源的复用。场景树（Scene Tree），则对场景具有组织作用，决定了场景的加载顺序和层次关系，也实现了场景之间的通信和关联

作为游戏开发工具，Godot当然具有资源化的特点。在项目结构与文件组织上，它具有清晰的结构分块，具有优秀的保存与加载机制，但同时也对开发者项目组织管理能力提出了更高的要求，以便实现高效运维。

2.3 学习资料

学习资料的丰富与否极大地影响了入门者的学习速度和热情。

Dev-C++自公布以来被广泛传播和使用，经过长时间的沉淀，积累了丰富学习资源。书籍方面，很多主流的C++学习教程将Dev-C++作为教学的标准软件进行示范，并且相关的中文原创和翻译书籍已经具有相当规模。关于在线资源，众多编程相关的网站，如CSDN等，囊括了Dev-C++从安装、使用到项目实例的一系列教程，初学者可以轻松获取获取相关信息。

相比之下，由于Godot的针对对象较为局限，且问世时间较短，它的纸质教程极其有限，并且大多为英文书籍，一定程度上造成了学习的障碍。同时，与Unity等主流游戏引擎相比Godot规模较小，吸引的优秀编程者的数量有限，它的课程等网络资源明显不足，对初学者查阅文档、相关社区信息检索等方面能力提出了更高的要求。

3 结论

在对Dev-C++和Godot这两款集成开发环境进行深入比较后，从初学者的角度出发，可以清晰地看到它们各自的特点和适用场景。

Dev-C++有着简洁的界面和专注于代码编写调试的多窗口布局，能让初学者迅速将精力集中于C++语言本身，快速熟悉编程基本操作。单文件开发功能极大降低了学习门槛，丰富的学习资料更是为初学者提供了充足的学习支持，有助于初学者快速掌握基础技能。然而，它在项目管理方面存在明显短板，复杂项目结构组织能力有限，库依赖处理不够便捷，开发功能上具有不足之处。 Godot作为更为现代化的IDE，其可视化场景编辑器便于图形化界面设计，信号机制增强代码独立性和扩展性等，使得程序开发更加高效灵活。项目管理采用场景树-场景-节点结构，促进游戏模块化开发和资源复用。但它对初学者的软件操作能力要求较高，无论是项目管理还是场景设计等都需要投入更多精力去学习掌握。而且，其学习资料相对匮乏，无论是纸质教程还是网络资源都不及Dev-C++丰富，这在一定程度上增加了学习难度。总的来说，Dev-C++适合那些希望先扎实掌握编程语言基础，逐步探索开发环境功能的初学者；而Godot则更吸引渴望快速上手游戏开发，愿意投入时间克服操作和学习资料不足困难，以充分利用其强大功能、提升个人项目开发水平的初学者。这两款IDE在编程学习和游戏开发领域都有着各自的价值，初学者可根据自身情况和需求进行选择。