EINSTIEG

Vor dem Abschluss und Export eures Games, müsst ihr zwangsläufig irgendwann nach Fehlern im Programmcode suchen. Das bleibt selbst den besten Programmieren nicht erspart. In diesem Kapitel erklären wir euch, wie ihr nach Fehlern suchen könnt und welche Handlungsempfehlung wir euch hierzu anraten

DEBUGGING

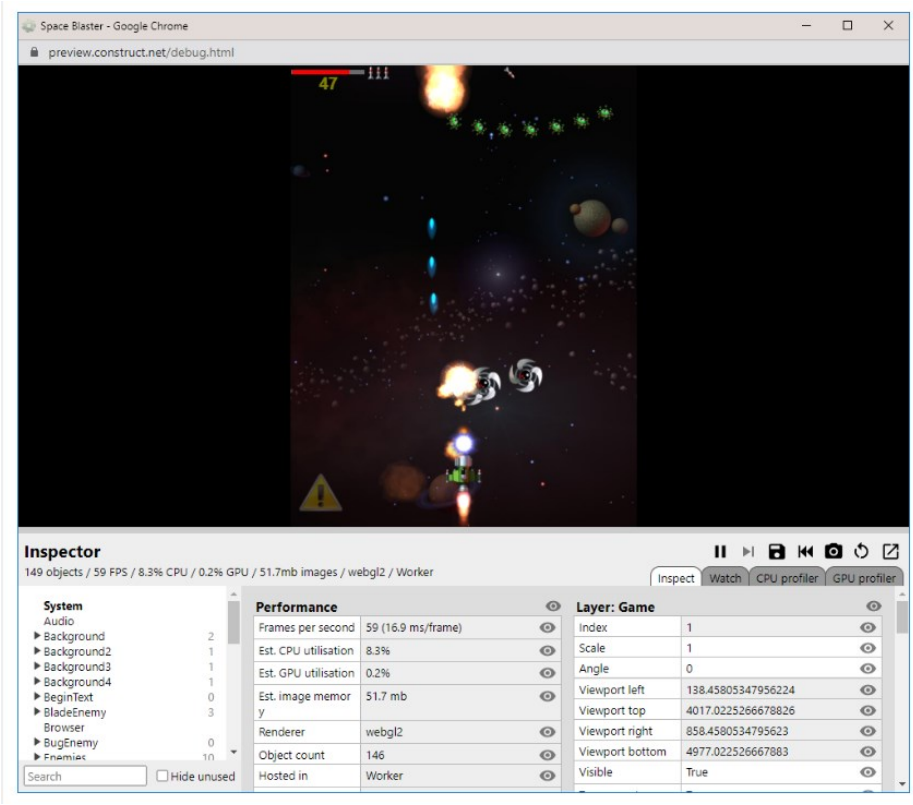
Bugs beziehen sich auf **Softwarefehler** – Dinge, die in Ihrem Projekt nicht so funktionieren, wie Sie es erwartet haben. Debuggen bezieht sich auf den Prozess der Behebung dieser Probleme. Der Debugger von Construct ist ein Tool, mit dem Sie Fehler in Ihrem Projekt finden und beheben können.

Der Debugger hat drei Registerkarten: Inspect , Watch, CPU-Profiler und GPU-Profiler

Ausführen des Debuggers

Der Debugger wird angezeigt, wenn Sie den Debug- Vorschaumodus auswählen. Dies kann über die Hauptsymbolleiste , das Hauptmenü , die Projektleiste oder über die Tastenkombination Ctrl+ erreicht werden F5.

Der Debugger funktioniert ähnlich wie eine gewöhnliche Vorschau , außer dass neben dem Projekt ein zusätzliches Fenster mit vielen Informationen und einigen Diagnosetools angezeigt wird.



Der Debugger zeigt Informationen über das laufende Projekt an

Anpassen des Debugger-Panels

Die Größe des Debug-Bereichs kann geändert werden, indem Sie den Größenänderungsrahmen oben entlang ziehen. Auf diese Weise können Sie es herausziehen, um weitere Informationen anzuzeigen, oder es auf seine Tools und eine Zusammenfassung der Leistungsinformationen reduzieren.

Der Debugger kann auch in einem eigenen Fenster angezeigt werden. Dies ist besonders nützlich bei Multi-Monitor-Setups. Das Projekt wird im vollständigen Browserfenster angezeigt, und ein separates Browserfenster zeigt das Debug-Fenster an. Klicken Sie dazu auf die Popout-Schaltfläche oben rechts im Debugger-Bedienfeld. Wenn Sie erneut darauf klicken oder das Popup-Fenster schließen, wird das Debugger-Bedienfeld wieder im Hauptfenster des Browsers angezeigt.

Haupt-Debugger-Befehle

Neben der Pop-out-Schaltfläche gibt es einige andere nützliche Tools. Sie sind wie folgt:

Pause: Pausieren Sie das Projekt, damit es nicht weiter fortschreitet. Dies ist nützlich, um eine Weile damit zu verbringen, einige Informationen zu einem bestimmten Zeitpunkt zu überprüfen. Wenn es angehalten wird, verwandelt es sich in eine Schaltfläche zum **Fortsetzen** ; Klicken Sie erneut darauf, um die Ausführung fortzusetzen.

Schritt kann nur verwendet werden, wenn er pausiert ist. Es rückt das Projekt um einen Frame vor. Die Delta-Zeit (dt) wird so eingestellt, als würde das Projekt mit 60 FPS laufen. Dies kann nützlich sein, um einen Moment Bild für Bild zu untersuchen und zu beobachten, wie ein Ereignis wie eine Kollision gehandhabt wird.

Speichern und Laden führen eine temporäre Speicherung durch, sodass Sie den Status des Projekts schnell speichern und diesen Status später jederzeit wiederherstellen können. Dies kann nützlich sein, um denselben Teil eines Projekts immer und immer wieder auszuführen. Der Status wird im lokalen Speicher des aktuellen Browsers gespeichert. Das Speichern ist nicht in einem anderen Browser verfügbar, aber auch nach dem Schließen und erneuten Öffnen, Neustarten usw. im selben Browser.

Screenshot erstellen lädt einen Screenshot der Hauptprojektansicht herunter und bietet ein nützliches Tool zum Erfassen von Bildern Ihres Projekts.

Neustart aktualisiert das Projekt und lädt es erneut von Grund auf neu.

Leistungszusammenfassung

Einige Details zur Leistung des Projekts erscheinen in der Haupttitelleiste des Debuggers und im Registerkartenbereich „Inspect“ für das Systemobjekt, das anfänglich angezeigt wird. Weitere Hinweise zur Leistung finden Sie unter Leistungstipps . Beachten Sie, dass der Debugger, da er viele Informationen anzeigt und verwaltet, selbst einen erheblichen Leistungsaufwand haben kann. Wenn Sie die Leistung messen, wechseln Sie am besten zu einem der Profiler-Tabs, oder verwenden Sie den normalen Vorschaumodus und zeigen Sie Leistungsmessungen mit Objekten an. Die Leistungsinformationen, die der Debugger anzeigt, umfassen Folgendes:

Die Objektanzahl (z . B. 500 Objekte): wie viele Objekte derzeit erstellt werden. Die Verwendung zu vieler Objekte kann die Leistung beeinträchtigen. Dieser Wert entspricht dem systemausdruck objectcount .

Die Framerate (z. B. 60 FPS): Wie viele Bilder pro Sekunde läuft das Projekt. Die Displays der meisten Systeme werden mit 60 Hz aktualisiert, sodass das Projekt für ein optimales Rendering mit 60 FPS ausgeführt werden sollte. Dieser Wert entspricht dem fps -Systemausdruck.

Die geschätzte CPU-Zeit (z. B. 20 % CPU): eine Schätzung, wie viel CPU-Zeit in der Logik des Projekts aufgewendet wird. Dies ist nicht immer genau, zumal es nur die für den Haupt-JavaScript-Thread aufgewendete Zeit berücksichtigt und nur als ungefähre Zahl betrachtet werden sollte. Der Profilerkann dies dahingehend aufschlüsseln, wie viel Zeit in jedem Bereich des Projekts aufgewendet wird, und wird später in diesem Handbuch ausführlicher beschrieben. Dieser Wert entspricht dem CPUUtilisation -Systemausdruck.

Die geschätzte GPU-Zeit (z. B. 20 % GPU): eine Schätzung, wie viel GPU-Zeit für das Rendern des Projekts aufgewendet wird. Dies ist auch eine Schätzung, die auf Hardware-Timern in der GPU basiert. Dieser Wert entspricht dem GPUUtilisation -Systemausdruck.

Die geschätzte Bildspeichernutzung (z. B. 32,5-MB-Bilder): eine Schätzung, wie viel Speicher von den aktuell geladenen Bildern im Projekt verwendet wird. Bilder verbrauchen normalerweise den meisten Speicher in einem Projekt, aber beachten Sie, dass dieser Wert alles andere ausschließt, wie z. B. Speicher, der zum Ausführen der Projektlogik oder zum Abspielen von Musik und Soundeffekten erforderlich ist. Weitere Informationen finden Sie in der Anleitung zur Speichernutzung . Dieser Wert entspricht dem Systemausdruck ImageMemoryUsage .

Einige zusätzliche Leistungsdetails werden im Abschnitt „ Leistung " der Inspektoransicht des Systemobjekts angezeigt, die standardmäßig angezeigt wird:

Collision checks/sec (zB 1144 (~22 / tick)): wie oft in der letzten Sekunde die Engine auf eine Kollision zwischen zwei Objekten testen musste. Kollisionsprüfungen werden durch die Sprite-Bedingungen „ Bei Kollision " oder „Ist überlappend" aufgerufen , und viele Verhalten führen automatisch zusätzliche Kollisionsprüfungen durch. In Klammern sind auch die durchschnittlichen Checks pro Tick angegeben. Wenn beispielsweise in der letzten Sekunde 600 Kollisionsprüfungen durchgeführt wurden und die Framerate 60 FPS beträgt, sind die geschätzten Prüfungen pro Tick 10. Dies sagt Ihnen, dass im Durchschnitt etwa zehn Kollisionsprüfungen pro Frame durchgeführt wurden, obwohl der tatsächliche Wert oft vom Frame abweicht -für Rahmen.

Poly-Prüfungen/Sek . (z. B. 60 (~1 / Tick)): Die meisten Kollisionsprüfungen sind sehr schnell, und die Engine kann trivial feststellen, dass sich zwei Objekte nicht überlappen (indem überprüft wird, ob sich ihre Begrenzungsrahmen nicht überlappen). Wenn sich jedoch die Begrenzungsrahmen von zwei Objekten überlappen, muss die Engine eine aufwendigere Prüfung durchführen, bei der die Kollisionspolygone jedes Objekts gegeneinander getestet werden. Dieser Wert gibt an, wie viele solcher Überprüfungen in der letzten Sekunde durchgeführt wurden, sowie den Durchschnitt pro Tick wie beim Wert Collision checks/sec . Normalerweise ist der Wert für Polychecks/Sek. erheblich kleiner, aber wenn er hoch ist, deutet dies auf ein mögliches Leistungsproblem hin.

Haltepunkte

Beim Ausführen des Debuggers ist es möglich, Haltepunkte zu setzen, um die Ausführung eines Ereignisblatts bei einem bestimmten Ereignis, einer bestimmten Bedingung oder Aktion anzuhalten. Weitere Informationen finden Sie im manuellen Eintrag zu Haltepunkten .

AUFGABE

Versucht nun selbst nach Fehlern in eurem Programmcode zu recherchieren. Unser Schulungsvideo begleitet euch zusätzlich durch diesen Prozess.