Mod SDK changelog

This is the changelog for the Core Keeper Mod SDK. It will contain changes to the SDK itself as well as internal changes to the project that isn't covered in the regular changelog.

It is still very much recommended to also read what is in the changelog for the Core Keeper application on Steam: https://store.steampowered.com/news/app/1621690?updates=true

0.7.4

- The mods that are loaded from mod.io will have an extra check that verifies the current version is supported by the mod, indicated by the Game Version tag. If this isn't set by the mod to the current version, it will still be possible to load the mod, but the player will get a warning first. This means it is more important to set this version tag correctly than before.
- The HealthChangeBuffer now has an entity field which indicates which entity is affected by the change and is stored in a singleton instead of on each entity.
- Expanded usage of the PugProperties system to avoid storing as much data on entities which instead can be shared. Each object entity now has an ObjectPropertiesCD component that can be used to look up shared data for that object. Added SetProperty functions to PugConverter where this data can be set.
- Some of the tile functions in Manager.multiMap have been replaced with Manager.multiMap.GetTileLayerLookup which gives you a job-compatible struct which can be used to access the same functions.

0.7.3

- You can choose a dedicated server install when choosing an install location in the SDK. Mods that are loaded via the SDK will also be detected properly by any connecting client even if the client uses the mod from mod.io. There is no automatic install of any missing mod that isn't installed via mod.io though.
- We are retiring the Translation component in favor of the LocalTransform component introduced in ECS 1.0. The main motivation for this change is to improve performance, as the Translation and LocalTransform must currently be kept in sync by copying the values back and forth. The Translation component will still be available in the project in order not to break serialized worlds, but no runtime entities are expected to have this component. This is one of the most commonly used components, but the following steps made the upgrade process fairly pain-free when we upgraded our own code:
  1. Use the following case-sensitive find-and-replace patterns. This took care of the vast majority of our usages.
     - `Translation translation` -> `LocalTransform transform`
     - `translation.Value` -> `transform.Position`
     - `, translation) in` -> `, transform) in`
     - `<Translation>` -> `<LocalTransform>`
     - `typeof(Translation)` -> `typeof(LocalTransform)`

2. Fix all resulting compiler errors. The most common cause is that the patterns don't catch variable names in some cases.
3. Replace all usages of `new Translation` with calls to `LocalTransform.FromPosition`.
4. Find and replace any remaining usages of the Translation component.

Unity's official ECS 1.0 upgrade guide might provide useful information on the new LocalTransform component:

https://docs.unity3d.com/Packages/com.unity.entities@1.0/manual/upgrade-guide.html#update-transforms-in-your-project.

- Completely removed the Rotation component, as rotation of entities is not used anywhere in Core Keeper. Setting the Rotation field on LocalTransform also has no effect.
- EntityMonoBehaviour is the base class for all of our graphical objects, and over the years it has grown to an unmanageable size as it has to support all our objects' specific use cases. We are making a push to return to a more component-based approach. Our aim is to make it less daunting and error-prone to add new graphical objects to the game. For this update specifically, we have:
  1. Added support for multiple IGraphicalObject components on the same object. All such components **at the object root** are picked up and have their methods invoked as appropriate.
  2. Added separate interfaces for components that only provide spawn/despawn behavior. Prefer using these when possible to avoid the overhead of empty GraphicalUpdate methods being called each frame.
  3. Added several new components that control aspects of graphical objects that were previously managed with settings on EntityMonoBehaviour. We encourage you to when possible use these components instead of the corresponding EntityMonoBehaviour settings. The old settings have not been removed, but they will not be developed further and may be removed at a future time. These components are listed below, for more information see their in-editor tooltips and how they are used on existing assets. `SpriteSkinFromPaintColor`, `SpriteGradientMapFromPaintColor`, `SpriteVariationFromEntityDirection`, `SpriteVariationFromEntityVariation`, `SpriteSkinFromEntityAndSeason`, `EnableEffectsForEntityVariation`, `OffsetFromEntityDirectionOrVariation`
- The hierarchies of a lot of our graphical objects have also had an overhaul to improve performance. In most cases this means porting an older object to use our in-house sprite rendering solution (SpriteObject). Apart from this we have also done lots of small changes to increase performance in various ways, see the performance section of the 0.7.3 changelog for the Core Keeper application for more information on this.

0.7.2

- Upgraded Unity version to 2022.3.10f1.
- Upgraded DOTS/ECS to 1.0. See links for upgrade instructions. The Translation component should still work as before, and any baking changes aren't relevant because we are using our own baking system (PugConversion).

- ○ https://docs.unity3d.com/Packages/com.unity.entities@1.0/manual/upgrade-guide.html
- ○ https://docs.unity3d.com/Packages/com.unity.netcode@1.0/manual/upgrade-guide.html
- ○ https://docs.unity3d.com/Packages/com.unity.physics@1.0/manual/upgrade-guide.html
- Burst version upgraded to 1.8.9.
- A lot of ECS systems have been converted to use ISystem instead of SystemBase to improve performance.
- Added an option for you to bypass the security check, adding a notice about it to the user: "Caution: 'Elevated Access' mods have increased access to resources outside the game like user files and internet. For the best experience, install only from reputable sources." The reflection API might be removed in the future in favor of using this option instead.
- The Unity.Collections package is embedded in the SDK, but the only change is that we have added the old FixedString32/64 types which are still used for some serialized components. Changing these to the new versions would have broken serialization.
- The DOTS version uses a very different serialization format for ECS worlds. As a result there is a separate application in CoreKeeper_Data/StreamingAssets/Patcher which is used to convert saves from before 0.7.2 to an interim format that the new version can read.