Draw.IO Technical assessment of the project

by K.Kostenkov for Voodoo

Table of contents

CPU

CPU spikes were detected in the following methods especially with high amount of entities in the game:

```
IAPlayer.Update
GameManager.Update
TerrainManager.FillCoroutine
DrawLine.Update
```

Deep dive investigation is recommended for the optimization.

RAM - Memory allocations

Allocates 18 Mb of memory. Consider optimizing

- `TerrainManager.Awake -> Resources.LoadAll<TerrainData>("Terrains"));`

Frequent memory allocations + excessive CPU usage

- `TerrainManager.FillCoroutine`
- `DrawLine.Update`

RAM - Memory leaks

Some scripts produce material instances and do not destroy them. This would produce memory leaks.

`PainSurface`

`BrushMainMenu`

`Brush`

`PowerUp_DeadMan`

GPU

Terrain chunks

- Are rendered with the standard material - simplify and consider moving to URP
- Swap the mesh for Quad instead of Plane

Camera

- Enable occlusion culling, investiate more if we could render what's not visible

GPU - Textures

Textures

- Fix max import size of textures in project
- Some sprites could be split and composed in runtime to reduce occupying space (ex. `Voodoo_Draw.io_Elements_GrandChampion3`)
- Some UI sprites could benefit of slice9 to reduce their size (ex. `Draw.IO_Picto_Perso_2`)
- Consider removing unsued textures (ex. `Ratings`)

Sprite Atlases

- Enable and pack UI textures (will fix NPOT problem)

GPU - Canvas

- Change to Screen Space overlay
- Get rid of layout groups where possible replacing them with anchors and manual UI composition
- Some UI elements are hidden by setting alpha to 0 (transition reasons). Assess the possibility to turn such game objects off.

GPU - Assets

- Disable receiving shadows at prefabs that don't need them
- Enable GPU instancing at materials
- Define the purpose of GameManager\Spotlight and turn if off if possible to batch more
- Assess an option to swap SkyDome for skybox that’s currently missing
- Create universal reusable materials for colors and assign them instead of changing colors manually. This will make brush parts batch more effectively

Build size

- Remove Unity splash screen (It is included in all projects by default even if custom splash is enabled)
- `Plan de travail 82`(1.3 Mb) texture is used in sky dome. Check business requirements for the size and necessity. (Refer to GPU - Assets section)
- Check optimization of `Vooddo_DrawGame_Io_Pinceau_3.fbx`model
- `EmojiOne.png`of TMPro is not used. Delete it

Build size - Resources

- `_MobileHapticManager` is in the `Resources` directory and in the game scene. Remove from resources.
- Pack `Brush data` and similar prefab containers to Addressables
- PostProcessing's `BlueNoise` is a large texture that is not used in the project
- Delete unused assets in general (that have missing scripts as well)
- Delete unused post-processing shaders from Resources in particular
- `Uber.shader` is not used. Delete it.

Architecture - Scene

- Move tips out of `Loading view` to a separate SO.
- Consider merging `Loading view` into the common `Canvas` and adding `Canvas Group` component
- Fix `Paint test` scene. Could be used to speed up the development
- Camera coordinates Game objects are unwieldy. Consider using `Vector3` serialized fields or separate Scriptable object containers

Architecture - Project

- Unity editor version is outdated and not supported anymore. Consider updating
- Switch project to IL2CPP and add ARM64 support
- Setup shades quality and distance in Quality settings
- Make sure haptic works on devices (missing native libs produce errors)
- Assess the option not to use animations in UI. Use tweening instead
- Move `GameViewUtils.cs` to `Editor` directory
- Remove mockup textures from version control
- Remove unused packages (including build-in ones) from Package manager

Architecture - Project - 3rd party

- `NavMeshComponents`directory duplicating built-in Unity lib. See what's not duplicated if anything and remove the rest
- Move all 3rd party packages to a common separate directory from Assets directory to make project navigation easier
- Assess present 3rd party packages. Some packages like `PolygonArsenal`could be either get rid of or partially deleted (by removing Demo assets and scenes). This will improve project navigation and project loading\import time
- Introduce assembly definitions for 3rd party packages that include code

Architecture - Code

- The project heavily `SingletonMB<MapManager>`Singletons uses. Dependency Injection framework introduction is advised
- Some `SingletonMB`(As well as some regular classes) have no reason to be MonoBehaviours. Remove unnecessary inheritance
- The project makes use of `enum`s. This will prevent project scalability and development Consider adding classes which behaviour will substitute `switch` clauses by inheritance
- Consider swapping `DefaultExecutionOrder`attributes with one class making ordered calls. This will make flow of control in the app easier to understand
- `GameManger`class is the god object. Deserves splitting into several classes.
- `PoolManager`class is present but is not used for object pooling. `FreePool` class presence is questionable
- `ListExtensions.Shuffle`effectiveness (and necessity) is questionable

Architecture - Code

- Get rid of preprocessor symbols where possible by implementing classes that implement common platform-independent interface. Otherwise refactoring and adding new supported platforms would be painful (ex. contents of `MobileHaptic` directory)
- `PlayerPrefs` class is used to store persistent data on the device. Encapsulate calls to it to be able to swap the implementation with save files later
- `StatManager` does not cache values. This makes it's usage I/O bound
- The project could benefit from UI manager of some sort that will cover the control of UI flow and make it more straightforward. Currently views are subscribing at their own will to game events. Introducing new UI flows could be difficult. Introduction of MV* pattern is advised
- Currently project's code style is far from the perfect state. Introducing code style convention and linter will bring more order and developer's happiness.

Architecture - Code - Data

- XP settings data is spread across `GamaManager`, `StatsManager` and `RankingManger`. Consider extracting and grouping settings to Scriptable objects hosted by a separate class
- Splitting `Constants` class is advised to separate gameplay settings from project-reserved names and make A\B testing easier in the future.

Bugs

- Continuing game on a device sometimes is not executed. Leads to a soft lock of the app.
- Shaking camera during the gameplay
- No way to select skin -1
- Missing sounds
- Not working native haptic libraries

Summary

The project is in a post-prototype state with most of the problems normally associated with prototypes made by beginner indie developers. Addressing the main issues would allow the project to be released in a soft launch followed up by further improvements.

Improving the current project rather than creating a new one by reusing assets and codebase looks more promising in terms of time, given the business requirement to get first results sooner.

Giving up on the issues of medium importance in medium term could make the project being rigid and error prone, reducing the speed of development and the results expected by business.