Sad Pumpkin Games

Turn-Based Game Engine

GSIP Proposal

Jake O’Connor

06.27.2021

Introduction

The *Sad Pumpkin Games Turn-Based Game Engine*, named after my theoretical indie game startup Sad Pumpkin Games, is an abstract, design-agnostic framework for the implementation of turn-based games such as JRPGs, Tactical RPGs, board games, and any other turn-based game. The design for the framework is heavily inspired by the works of linguist James Paul Gee, specifically his theory of Unified Discourse Analysis, which aims to abstract the language of video games down to acts of turn-taking like other forms of discourse. The *SPG Turn-Based Game Engine* abstracts the turn-taking logic of turn-based games down to their shared foundations, which allows it to be used as the basis for any kind of game in which turn resolution is required.

Innovation Claim

The *Sad Pumpkin Games Turn-Based Game Engine* is innovative in that it is a design-agnostic and platform-agnostic abstraction of turn-based game execution. Unlike other products on the market, this engine has no inherent biases towards specific genres or platforms, it can support a tactical RPG written in Unity or a basic chess app written in ASP.NET, or anything in between.

Research into Prior Art

There are countless other products in the market that provide varying levels of support for turn-based games. The downsides to most of these products, in my opinion, are that they fall into one of two categories: those which are too restrictive on design, and those which are too restrictive on platform. There is a plethora of RPG, JRPG, and Tactical RPG toolkits in the market through places like the Unity Asset Store, but these toolkits are generally design-restrictive, requiring that the game being made follows the design standards of the third-party plugin's creator. There are also independent solutions,

such as the RPG Maker series of tools, but these solutions limit both the design space and the release platforms available for users.

Turn Based Strategy Framework by Crooked Head, available on the Unity Asset Store, is a toolkit and template for users to implement turn base strategy games or tactical RPGs within Unity (Crooked Head). This product has helped multiple game teams bring games within these genres to market successfully. This product is similar to the *SPG Turn-Based Game Engine* in that it assists users in creating customizable turn-based gameplay. There are major differences between the two toolkits though, most notably that *Turn Based Strategy Framework* is built atop Unity and cannot run independently, making it unusable for products not developed within the Unity ecosystem. Additionally, this framework specifically focuses on the development of turn-based strategy and tactical role-playing game genres, versus being a more general, abstracted toolkit.

Turn-based RPG Battle Engine 2D by ByteVeil, available on the Unity Asset Store, is a toolkit and template for users to implement 2D turn-based RPG battles within Unity (ByteVeil). This product is designed as a jumping off point for game developers, and by the author's admission requires significant engineering in order to create gameplay with. This product is similar to the *SPG Turn-Based Game Engine* in that it offers users a foundational framework which they need build upon to create their own concepts. Where it differs from the *SPG Turn-Based Game Engine* is that it is built atop the Unity engine, relying on Unity-specific object types, animation components, and editor windows in order to edit and customize, making it unsuitable for use in any platform outside of Unity.

ORK Framework by Gaming Is Love, available both through the ORK website and the Unity Asset Store, is a game framework and toolkit for developing RPGs of all types, including turn-based, active time, and real time (Gamingislove). This product is a holistic framework which offers developers a near code-less ability to develop entire RPG products from scratch within Unity. This product shares almost

no similarities with the *SPG Turn-Based Game Engine* other than their exposure of customizable turn-based behavior to the user. They are different in every other way, where the *ORK Framework* is more of an RPG toolkit that includes things like crafting, events, character classes, and the like, the *SPG Turn-Based Game Engine* is fully design-abstracted, not requiring any specific design-level implementation details.

Marketplace Analysis

The *Sad Pumpkin Games Turn-Based Game Engine* is being developed as an in-house solution for indie game development at Sad Pumpkin Games in order to speed up iteration time on new game designs. Due to the inherent abstraction of the engine, it can be adopted to a vast array of design-specific implementations without having to reinvent the wheel or work against the engine, unlike when using products that are design-constrictive. The engine will be freely available as a DLL for anyone who wishes to use it as a foundation to develop their own turn-based games without having to reimplement the foundational turn-taking logic each time. Releasing the engine as a DLL built on .NET makes it easily usable by those making games in any .NET-based framework (such as Unity, ASP.NET, or WPF) as well as those making games within other VM languages (such as Java).

Materials and Methods

The *Sad Pumpkin Games Turn-Based Game Engine* is developed and written in C# using the .NET Core framework. .NET Core's portability makes it ideal for an abstracted engine, as the resulting DLLs can be used as-is within other .NET products (Unity, ASP.NET, WPF, etc.), via simple tools within other VM languages (Java, etc.), or via a wrapper DLL in more base languages such as C++. .NET Core's ubiquitous adoption also ensures that in almost all cases users of the engine will not need any extra software or packages in order to being working with it.

The code for the engine itself is designed in such a way that it makes use of many common software engineering patterns, especially those used within the games industry. This further helps with the startup time when beginning to use the engine, as the common patterns should make it simpler to understand and use. One such pattern is the Façade Pattern, which is heavily used internally within the engine and guarantees that regardless of design-level implementation decisions by users, the engine will operate based off the interfaces (facades) that are required.

Timeline

Development of the *Sad Pumpkin Games Turn-Based Game Engine* began during the Fall semester of 2020 in MS587. Originally, the engine was just a convenient way of separating the business logic from the view logic of that course's final project, but after reading and discussing James Paul Gee's theory of Unified Discourse Analysis during MS504 I began to contemplate the wide application of a fully abstracted turn-based engine for use in games across genres. After selecting the idea of an abstract turn-based engine as my GSIP in the beginning of 2021 development began, albeit very slowly. The original plan began as the development of the engine plus three prototype games built upon it in different genres, but as time progressed that seems to become less and less likely to be possible in the time provided. The current plan for a timeline is to cobble together as much provable use as possible in the remaining time, which is difficult for something that is purely algorithms and abstractions.

First Iteration

The very first iteration of the *Sad Pumpkin Games Turn-Based Game Engine* exists within the business logic of a ASP.NET game prototype called *Thirty Day Hero*. The game can be played online at http://30dh.sadpumpkin.com/ and uses the prototype turn-based engine to run all the core combat

logic within the game. Since this prototype, the engine has received at least a dozen major overhauls to further abstract its functionality and facilitate more customizable uses.

References

Abstract Factory. Refactoring.Guru. (n.d.). https://refactoring.guru/design-patterns/abstract-factory.

Adapter. Refactoring.Guru. (n.d.). https://refactoring.guru/design-patterns/adapter.

Arktentrion. (n.d.). *2D RPG Kit*. Packs | Unity Asset Store. https://assetstore.unity.com/packages/templates/packs/2d-rpg-kit-163910.

ByteVeil. (n.d.). *Turn-based RPG Battle Engine 2D*. Game Toolkits | Unity Asset Store. https://assetstore.unity.com/packages/tools/game-toolkits/turn-based-rpg-battle-engine-2d-135496.

Chain of Responsibility. Refactoring.Guru. (n.d.). https://refactoring.guru/design-patterns/chain-of-responsibility.

Crooked Head. (n.d.). *Turn Based Strategy Framework*. Systems | Unity Asset Store. https://assetstore.unity.com/packages/templates/systems/turn-based-strategy-framework-50282.

Dextero Solutions. (n.d.). *RPG All-in-One*. Systems | Unity Asset Store. https://assetstore.unity.com/packages/templates/systems/rpg-all-in-one-53542.

EviLA's Unity Assets. (n.d.). *EviLA's RPG Pack For Invector*. Systems | Unity Asset Store. https://assetstore.unity.com/packages/templates/systems/evila-s-rpg-pack-for-invector-102817.

Facade. Refactoring.Guru. (n.d.). https://refactoring.guru/design-patterns/facade.

Gamingislove. (2019, January 30). *Features*. ORK Framework - The complete RPG Engine for Unity. http://orkframework.com/features/.

Gee, J. P. (2015). *Unified discourse analysis: Language, reality, virtual worlds, and video games*. Routledge.

Prototype. Refactoring.Guru. (n.d.). https://refactoring.guru/design-patterns/prototype.

RPG Maker MZ: RPG Maker: Make Your Own Video Games! RPG Maker MZ | RPG Maker | Make Your Own Video Games! (n.d.). https://www.rpgmakerweb.com/products/rpg-maker-mz.

Strategy. Refactoring.Guru. (n.d.). https://refactoring.guru/design-patterns/strategy.