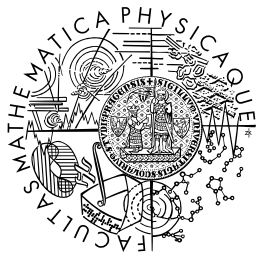

FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

BACHELOR THESIS

Miroslav Valach

Diplomacy-Based Strategy Game

Department of Theoretical Computer Science and Mathematical Logic

Supervisor of the bachelor thesis:	Mgr. Martin Pilát, Ph.D.
Study programme:	Computer Science
Study branch:	System Programming

Prague 2023

I declare that I carried out this bachelor thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Author's signature

I dedicate this thesis to my godchildren, whose presence and joy have been a constant inspiration. Additionally, I would like to express my gratitude to my supervisor, Mgr. Martin Pilát, Ph.D., for invaluable guidance. Finally, I extend my thanks to my family and friends for supporting me throughout my studies.

Title: Diplomacy-Based Strategy Game

Author: Miroslav Valach

Department: Department of Theoretical Computer Science and Mathematical Logic

Supervisor: Mgr. Martin Pilát, Ph.D., Department of Theoretical Computer Science and Mathematical Logic

Abstract: Strategy games are known for allowing players to choose from a vast array of different strategies that can be employed to achieve victory. The majority of these games revolve around the standardized pillars of 4X (Explore, Expand, Exploit, Exterminate) games such as Civilization or Stellaris. However, these pillars often encourage conquest or aggressive means to achieve victory, thereby rendering a peaceful approach as a rarely viable strategy to pursue.

In this project, our aim was to create a strategy game that focuses on player interaction through diplomacy. The main goal is to provide players with freedom similar to Diplomacy, where players can and have to utilize multilateral politics in order to achieve victory.

As a proof of concept, we have successfully developed a game prototype using Unreal Engine. The prototype showcases a diplomacy-based game with multilateral diplomacy at its core. The gameplay demonstrates the viability of diplomacy as the primary strategy for achieving victory in video games.

Keywords: strategy game, diplomacy, multilateral politics, multiplayer, diplomatic victory

Contents

Introduction

In this thesis, we seek to explore and create a prototype for a strategy game focused on diplomacy. The interaction between players can empower the design of strategy games. This allows players to express themselves in a way that is not present in standard economic and military approaches. A secondary goal is to create an API for artificial intelligence that enables players to utilize the system with and against an AI opponent.

Motivation

Strategy games are diverse and each focuses on different type of strategies. From city-builders to grand strategy games. Diplomacy is an aspect that these games often lack or is contained as an afterthought. Even the common acronym 4X stands only for Explore, Expand, Exploit and Exterminate. Games such as *Sid Meier's Civilization® VI* [1], *Europa Universalis IV* [2], *Stellaris* [3] and others fail to make diplomacy equal to other aspects. In most cases, extermination in the form of conquest is always preferred and in many cases it is the optimal choice. The usefulness of diplomacy is very limited, and upholding or breaking a deal is often inconsequential. Furthermore, all other game mechanics overshadow the impact of diplomacy on the game's outcome.

Goal

As a part of this thesis, we aim to create a prototype for a multiplayer strategy game. This prototype should distinguish itself by being designed with diplomacy in mind. Various mechanics and design decisions are expected to actively encourage players to engage in diplomacy. At the same time, we want to make it an engaging and fun part of the core gameloop. The game should be playable against artificial intelligence. Diplomatic mechanics need to be well defined as part of the API.

Structure

In the first chapter, we analyze diplomatic mechanics across various strategy games in different sub-genres and mediums. The second chapter explains design behind our prototype. That is, how to design a game with diplomacy as the core of its identity. The third chapter covers implementation of the prototype. The fourth chapter describes how to use the prototype. Finally, the fifth chapter summarizes the results and explores ways to expand upon them.

Chapter 1

Analysis

In this section, we take a closer look at a few games. They serve as the primary source of inspiration for this thesis. We keep the analysis short by focusing only on diplomatic interactions between players. Gathering resources, building structures, combat and similar interactions are not covered. Our diplomacy model should work with most common designs of these features. Games take different approaches to designing diplomacy and player interaction. We are focused on creating a strategy game in the spirit of Grand Strategy Games (GSG) like *Stellaris* [3]. Traditional board games and card games are analyzed as well.

1.1 Terminology

Definition of diplomacy can be found in *The Palgrave Macmillan Dictionary of Diplomacy* [4]:

> *The conduct of relations between *sovereign states through the medium of officials based at home or abroad, the latter being either members of their state's *diplomatic service or *temporary diplomats.....Diplomacy is therefore the principal means by which states communicate with each other, enabling them to have regular and complex relations.*

This definition is also explored in "Digitising Diplomacy: Grand Strategy Video Games as an Introductory Tool for Learning Diplomacy and International Relations" [5], where it is used in terms of strategy games.

In terms of our game, players control factions, acting as diplomats. The communication is achieved through player interactions within the game mechanics.

1.2 Real-Time Strategy games

Common examples of Real-Time Strategy (RTS) games include *Age of Empires IV* [6], *StarCraft II* [7], *Warcraft III: Reforged* [8] and *The Lord of the Rings: The Battle for Middle-earth* [9]. The definition of RTS as a genre is subjective. In general it covers all strategy games, where players act simultaneously in real-time.

The shared focus of aforementioned games is fast-paced playstyle in a 1v1 gamemode, where two players face each other. Often, the focus is on player expression through mechanical skill or simply maximizing effective Actions Per Minute (APM). Match results depend more on APM than on the underlying strategy. Especially for players of similar skill. Matches commonly take less then an hour.

1.2.1 Age of Empires IV

We chose Age of Empires IV due to its popularity and author's familiarity with the game. Age of Empires IV shares diplomacy and tributes mechanics with other entries in the series. With author's popular mod *Advanced Game Settings* [10], diplomacy is covered in more extensive manner over the previous entries. Diplomacy in the Age of Empires IV can be split between the Tributes and Relations as seen in Figure 1.1.

Sending tributes is a mechanic that enables player to send resources to another player. Each transaction is taxed. Changing relations is a mechanic that enables players to switch attitude towards other players. Community refers to relations as "Diplomacy". The available options are: friendly, neutral or enemy. A player can view other players' attitudes towards them and manage their own attitude towards other players. An **enemy** player is attacked automatically by your units. A **neutral** player can be attacked only manually. A **friendly** player can not be attacked and their units can pass through your gates.

Players can choose to either play Free For All (FFA), each player against all other players, or align themselves into static teams before the game begins. Players in a team are mutually friendly. The other option is to have dynamic teams. Any group of players is automatically counted as a team, if all members of the group are mutually friendly. If at least one player triggers a victory condition, his team will achieve victory. Relations and dynamic teams are part of the aforementioned mod.

The game has three victory conditions:

- **Landmarks** - destroy all enemy landmarks

- **Sacred Sites** - hold and defend map objectives
- **Wonder** - build and defend wonder

Neither of these victory conditions directly requires diplomacy to achieve victory. As players cannot share victory, using relations for any purpose other than undermining other players is counter-intuitive. Thus without dynamic teams the use of relations in a positive sense is discouraged. Dynamic teams do not resolve this issue, as winning member of the team can always leave the team and achieve victory alone. Static teams do not encourage any form of diplomacy, except for optimal distribution of resources within the team.

Figure 1.1 Diplomacy in Age of Empires IV, provided by *Advanced Game Settings* [10].

1.3 Turn-Based Strategy games

Common examples of Turn-Based Strategy (TBS) games include *Sid Meier's Civilization® VI* [1], *Old World* [11], *HUMANKIND™* [12], *Age of Wonders IV* [13] and *Heroes® of Might and Magic® III HD* [14]. In general, it covers strategy games, where gameplay is split into turns. These turns can happen simultaneously or sequentially between players. Very often, strategy turn-based games are synonymous with 4X[1] strategy games.

[1]Explore, Expand, Exploit and Exterminate

The 4X subgenre is based on four pillars:

- **Exploration** - focuses on discovery of new areas on the game map
- **Expansion** - occupying as much space on the map as possible
- **Exploitation** - taking advantage of owned resources in various forms and using them in an efficient way
- **Extermination** - waging conflict against other players and removing them from a game

These games are mostly the opposite of RTS games. Matches are played against multiple opponents and take tens to hundreds of hours to finish. Strategy and optimization play greater role than APM and related mechanical skill. The player is presented with multitude of victory conditions, each supporting a different playstyle.

1.3.1 Sid Meier's Civilization VI

We chose Civilization VI as the best known representative of 4X games. Civilization offers players an opportunity to engage in both bilateral and multilateral diplomacy.

Bilateral diplomacy has two forms. The first one is a leader interaction, where player can take direct action against another player. The second is making a deal or a demand in a system called The Bargaining Table. Both are accessed through the leader interaction screen as seen in Figure 1.3a. The player can select items they request or demand from the target and what they are willing to offer in exchange. Items include resources, various types of alliances and other. This can be seen in Figure 1.3b.

The lead designer of the previous Civilization games and recently of Old World mentioned in *My Elephant in the Room: An 'Old World' Postmortem* [15] that The Bargaining Table was a mistake and defines it as one of the cursed problems as described by Jaffe [16]. It crates a disconnect between the power and the flexibility of real diplomacy, where personality flaws come into play, and the simplified system of The Bargaining Table. In real diplomacy offending the other side generally ends the process and has consequences. In Civilization the consequences are meaningless as you can restart the process immediately.

Multilateral diplomacy is in the form of a voting screen. This is part of the The World Congress mechanic. All players can vote in favor or against certain resolutions that impact the specified player or all players. The World Congress allows players to access the diplomatic victory condition. The result of a single session is shown on Figure 1.2.

World Congress is intertwined with The Bargaining Table as the player is required to obtain diplomatic favor to force his goals in a vote. As we mentioned before, the table is flawed, so it is very easy to abuse the table to gain favors from other players.

From our point of view, diplomacy as a mechanic in Civilization lacks impact and consequences. Diplomacy can be safely ignored as other mechanics have a bigger role in determining the outcome of the game. Wide play[2] mitigates any benefits a player could receive from diplomacy with other players. Sharing resources is not as valuable as controlling them directly. Wide play can be also achieved by conquest, which makes it the superior choice as you eliminate rivals.

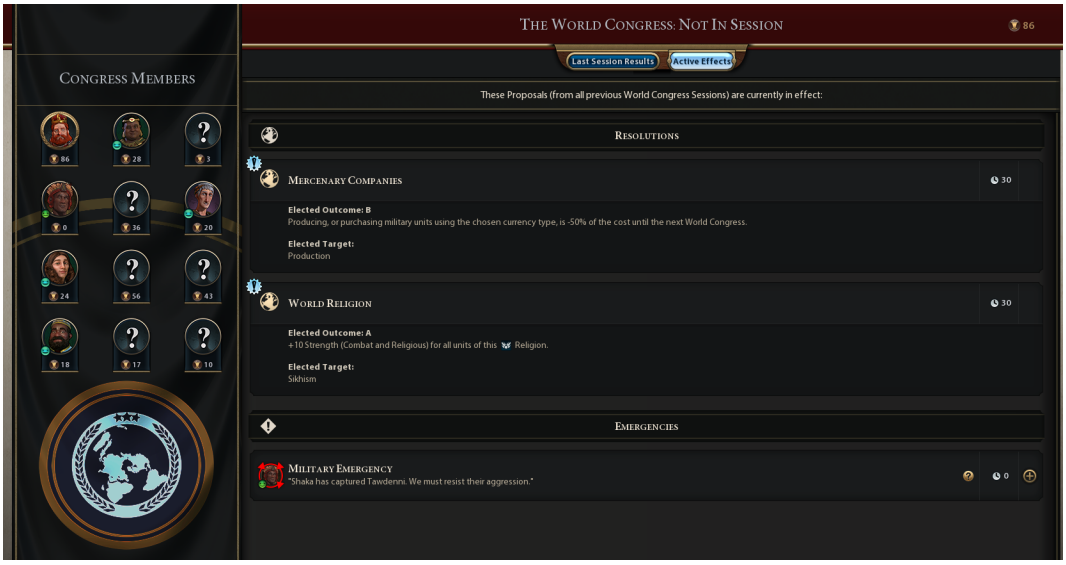

Figure 1.2 The World Congress in Civilization VI

[2] strategy focused on expansion of your land

(a) The Leader Interaction in Civilization VI

(b) The Bargaining Table in Civilization VI

Figure 1.3 Bilateral diplomacy in the Civilization VI.

1.4 Grand Strategy games

Common examples of Grand Strategy games include *Crusader Kings III* [17], *Europa Universalis IV* [2], *Hearts of Iron IV* [18] and *Victoria 3* [19].

The aforementioned games share common aspects. For example: *Real-time gameplay* with active pause. The map is based on a *historical state* of our world. The map is used for visualized data and relations. Games usually contain *hundreds of players or AIs* controlling political entities. Game mechanics are usually complex and require deeper understanding to make effective use of them. Gameplay, including warfare, is done in a more abstract manner as the player controls the whole political entity. The victory condition is often absent. The player is left to define his own goals over the course of the play-through. The only way to lose is to be completely conquered.

1.4.1 Europa Universalis IV

We choose Europa Universalis IV for further analysis of diplomacy in grand strategy games. Based on our experience, Europa Univeralis IV has the greatest focus on diplomacy. This game was also chosen as a subject in "Digitising Diplomacy: Grand Strategy Video Games as an Introductory Tool for Learning Diplomacy and International Relations"[5]

Europa Universalis IV covers primarily bilateral diplomacy in the form of diplomatic interactions. These can be split between requests, offers, demands and gifts. Interactions commonly result in relations such as alliances, vassalisations, political marriages, military accesses and others. Relations have various effects each with different consequences, if they are not upheld. Figure 1.4 illustrates various options available to a player and Figure 1.5 shows current relations of the player.

In declarations of both war and peace, warfare incorporates multiple entities based on existing alliances. Peace deals as seen in Figure 1.6 have the form of a one-sided bargaining table similar to the one in Figure 1.3b.

Multilateral diplomacy is present in a very limited form. Most prominent example would be Holy Roman Empire and the associated selection of the next Holy Roman Emperor. In this mechanic, current electors can vote for any eligible nation to inherit the emperorship once the current ruler perishes as seen in Figure 1.7.

Diplomacy in Europa Universalis IV generally serves as a stepping stone for conquest. Although player goals are not specified, players usually fall back on conquest as either the most intuitive goal or the necessary step to achieve other goals. Because of this, diplomacy in general serves only as a tool for further conquest or as a defense against being conquered.

Figure 1.4 Diplomacy Actions in Europa Universalis IV

Figure 1.5 Diplomacy Overview in Europa Universalis IV

Figure 1.6 Peace deal in Europa Universalis IV

Figure 1.7 HRE in Europa Universalis IV

1.4.2 In-between genres

We can find games that stand between different genres such as *Total War: WARHAMMER III* [20] or *Stellaris* [3]. In terms of diplomacy, however, they do not fare any better than the previously mentioned titles.

For example, Stellaris is associated with both the 4X and the GSG genres. Stellaris fully covers all four pillars of the 4X genre and combines it with mechanical depth of GSG intertwined systems. But the additional complexity serves very little to no advantage in encouraging diplomacy. In Figure 1.8 we can see the Galactic Community mechanic similar to The World Congress mentioned in Figure 1.2. Galactic Community provides an option to shape galactic rules affecting all members of the galaxy. Resolutions passed by the Galactic Community have usually no immediate payoff and have minimal effect on the player during the play-through. This often creates situations, where players barely interact with the system.

In conclusion, while Stellaris offers a higher degree of flexibility, its impact on diplomacy is not significantly positive. The presence of various other mechanics tends to overshadow the importance of diplomacy within the game.

Figure 1.8 Galactic Community in Stellaris

1.5 Board games

Most board games rarely have any form of diplomacy. In many cases interaction between players is not defined by the game rules. In games that have rules for interactions between players, it's heavily restricted. For example, by being limited to drawn cards such as in *Terraforming Mars* [21]. Other games impose rules that might not always be intuitive. *The Settlers of Catan* [22] allows trading only with the player that is currently playing.

1.5.1 Diplomacy

Diplomacy[23] is a board game created by Calhamer in 1959.

Game rules of Diplomacy are simple. In the negotiation phase players bargain over which actions they should take. Actions are executed in the movement phase. Players can lie during the negotiation phase and perform differently during the movement phase.

The key takeaway is that player actions are simple and do not overshadow diplomacy in the negotiation phase. This means that social interaction, interpersonal skills and overall strategy determines the winner. Resources of a single player are often not enough to achieve anything on their own.

In comparison, video games have much more complex mechanics. These mechanics occupy most of the player's attention and are entertaining on their own without having to engage in diplomacy. Consequently, players are not reliant on other players in achieving their goals. This implies that players do not feel an immediate need to hold each other accountable for deception. Thus, diplomacy often ends up as the least important mechanic.

1.6 Card games

One unlikely game, where we can find an important element required for functional diplomacy is *Marvel Snap* [24]. This collectible card game takes inspiration from the classic card game Poker as is stated in *Designing 'MARVEL SNAP'* [25].

1.6.1 Marvel Snap

This game is played as duel of two players that wager currency called *cubes* on their victory. The card mechanics of the game itself are not important for

understanding the mechanic we are interested in:

1. The game starts with forced bet of one cube by both players.
2. In the final turn of the game, the bets are doubled.
3. At any point during the game each player can *snap* once. Each *snap* doubles both players' bet at the start of the next turn.
4. A player can retreat at any point, losing their current bet, but mitigating the loss of additional cubes.

This unique mechanic distinguishes it from other collectible card games. The cubes are desirable to advance the meta progression of the game.

"Snapping" in the game usually implies being in the winning position. This puts pressure on the other player to determine whether it is a bluff and to evaluate the risks. In a disadvantageous position the player can retreat to mitigate his losses. This makes bluffing also a viable tool to win a game, making the other player retreat from his winning position.

Strategy videogames often do not present viable opportunities for deception. In order to bluff, the game needs to be designed with this in mind. As we learned in the game *Diplomacy* 1.5.1, it is a vital part of diplomacy in general.

Chapter 2

Game Design

In this chapter, we discuss game design for creating *Koruna* – a multiplayer diplomacy-based strategy game. As mentioned in the Analysis chapter, most current video games fail to implement diplomacy as a core mechanic. Consequently, other game mechanics, such as warfare or economic management tend to overshadow the role of diplomacy.

Game design goals:

1. Diplomacy is a core mechanic.
2. Diplomacy contains elements of espionage and deception.
3. Diplomacy is led by both bilateral and multilateral interactions.
4. Multilateral interactions have more impact than bilateral interactions.
5. The game is extendable and open to inclusion of additional mechanics, such as the ones commonly found in strategy games.
6. The game can be played by AI players through the provided API.

In the following sections, we explain and argue for the game's design decisions. These were made to uphold the game design goals.

2.1 Player Count

Multilateral diplomacy only makes sense with at least three players. Multiplayer strategy games can involve up to hundreds of players. Thus we do not put a limit for the maximum players involved. Most of the strategy games in competitive

settings involve two to eight players. Too many players may make the game overwhelming to play. Therefore, eight players seems to be a reasonable compromise for an expected and suggested amount of players as our game is competitive in nature.

2.2 Timekeeping

Player activity in the game can be handled in two major ways. Turn-based approach separates player activity into turns. Real-time approach allows players to act continuously and without interruption. There are examples of mixed approaches in various games. For example Europa Universalis IV uses real-time approach with an active pause mechanic. Active pause means you can queue up action during the paused state. Once the game is unpaused, the actions are executed.

For social interaction to play a bigger role in the game, players need space for discussion. Real-time games pressure players to focus on the highest priority tasks. This inhibits the development of diplomacy as a mechanic. In other words, players need time to talk and analyze the situation. Turn-based approach naturally allows allocation of time for social interaction. However, the turn-based approach results in slow and sluggish gameplay. Therefore, we start at a real-time approach. Then we modify it to accommodate the needs of diplomacy-driven gameplay. This leads us into combining the turn-based and the real-time approach.

The gameplay is separated into turns. However, during each turn all players act simultaneously in real-time. Turns are important as each turn allows for only one deal to be created. For the turn to end, each player has to indicate that they are ready, by using the *finish turn* action. This creates space for negotiations as players that are participating in the deal can postpone the next turn. Only in the new turn are players attributed additional resources to perform more actions and a new deal can be created. Figure 2.1 demonstrates the flow of the game.

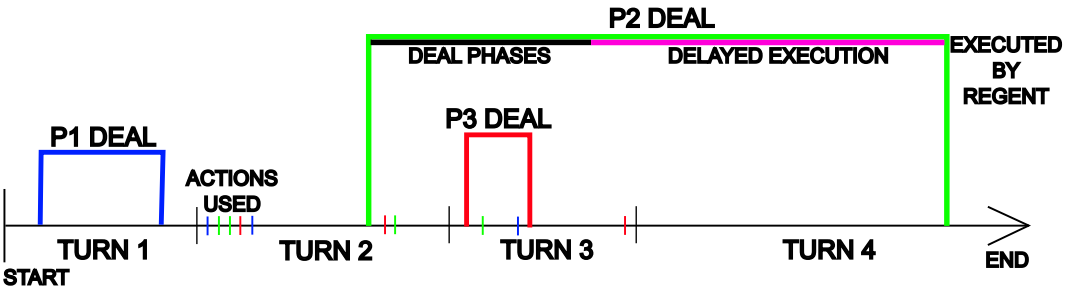

Figure 2.1 Timeline showcasing the flow of the game. Actions are available for use at any point during the turn as seen in Turn 2 and Turn 3. Deals can be contained either within a single turn or be a part of multiple turns as is shown by P2 Deal that was initiated by the second player.

2.3 Game length

Availability of players greatly influences length of individual matches in multiplayer environments. Shorter length of matches makes the game more approachable. At the same time, very long matches are rarely preferred in competitive settings, which often appear around multiplayer games.

With this in mind, we have decided to focus the game on the shorter end of the game length spectrum. However, diplomacy is a lengthy process and predicting the exact an amount of time is difficult. Regardless, we still aim for approximately one hour long matches that should easily fit into a single session.

2.4 Actions

There are multiple problems related to actions in the previously mentioned games.

1. Diplomacy is not a requirement of victory conditions.
2. Actions are the more optimal way to reach victory.
3. Diplomacy lacks payoff.
4. Deception and subterfuge against other players lacks consequences.

In our game the actions that immediately impact the game world are called **stratagems**. They consist of either direct interaction with the world (take land, exploit land) or bilateral diplomacy (give land, give resources). Stratagems cost a finite amount of resources to execute. Our final list of stratagems and description

of their effects can be found in Table 4.2 and Table 4.3.

We choose to design stratagems so they don't have a lot of power by themselves. Additionally, players are limited by the amount of available stratagems they can use during the match. Each player can select only a few stratagems from all available stratagems before the start of the match. This is akin to building a deck in deck building card games. To access additional stratagems they need to engage and interact with other players during the game.

These limitations placed on stratagems ensure that they are not the optimal way to reach victory. Players need to engage in diplomacy to take an advantageous position. Consequently, this ensures that diplomacy has a proper payoff.

The need for diplomacy also solves the problem of subterfuge. For example, sabotaging a deal has both positive and negative consequences. The immediate result is beneficial, as player gains an advantage over the opponents. But the gradual result is mostly negative, as trust issues will interfere with the smoothness of future deals. Other players will be less likely to include the player in the future deals.

Stratagems play a role in determining the initial strategy, how each player approaches the match. As a result stratagems feel impactful and shape various strategies.

Besides stratagems, players can engage in multilateral diplomacy. This mechanic is called a **Deal** and is discussed in the Deals subsection of the next section.

2.5 Diplomacy

Bilateral diplomacy is done by what we call decision actions: interactions, offers, requests and demands.

Multilateral diplomacy is covered by deals. Deals also have an important role in communication, as it is the only place where players can communicate via text messages.

2.5.1 Decisions

Some stratagems provide us with decision actions. These further subdivide into:

- Interactions - A player can directly affect the target player. That player does not get a say in this.
- Offers - A player can offer to use his stratagem on the target player. That player can either accept or decline the offer.
- Requests - A player can request another player to use their stratagem on the requester. That player can accept or decline without immediate consequences.
- Demands - A player can demand another player to use their stratagem on the requester. That player can accept or decline. However, declining can have automatic consequence based on other stratagems such as Burglary.

2.5.2 Deals

Each deal is composed out of multiple phases. Each phase serves a specific purpose and aims to simplify the potential back-and-forth that would happen in unregulated discussion. Thus structure of deals aims to prevent stalemates in negotiations. Flow of the deal can be seen in a Figure 2.2.

These are the phases of a deal:

1. Assembly of participants.
2. Definition of discussion points.
3. The vote to accept or refuse the deal.
4. Resolution of the points.
5. The deal execution by the regent.

These phases are implemented in following manner.

Only one player can make a single deal each turn. This player is referred to as the **regent**. When the regent makes a deal, they can choose which players are invited. Players can reject this offer. The position of the regent is rotated among the players across the game turns.

Each deal consists of points. A point describes a single usage of a stratagem by a participant. Any participant can create any number of custom points. Players can specify any stratagem from the collective pool of all stratagems belonging to

all players participating in the deal.

Once the points are made, the regent can start a vote for the execution of the deal. Participants can either accept or reject. If any participant rejects, the deal fails. Otherwise the deal passes.

If the deal passes, each participant has a choice of how to approach the points. The actors of each point can choose to accept, refuse or alter the point. Each participant can additionally choose to sabotage the points of other actors. Once each participant finishes his response, the regent can execute the deal either immediately or in a later turn. Once the deal is executed all participants receive some amount of the reputation resource.

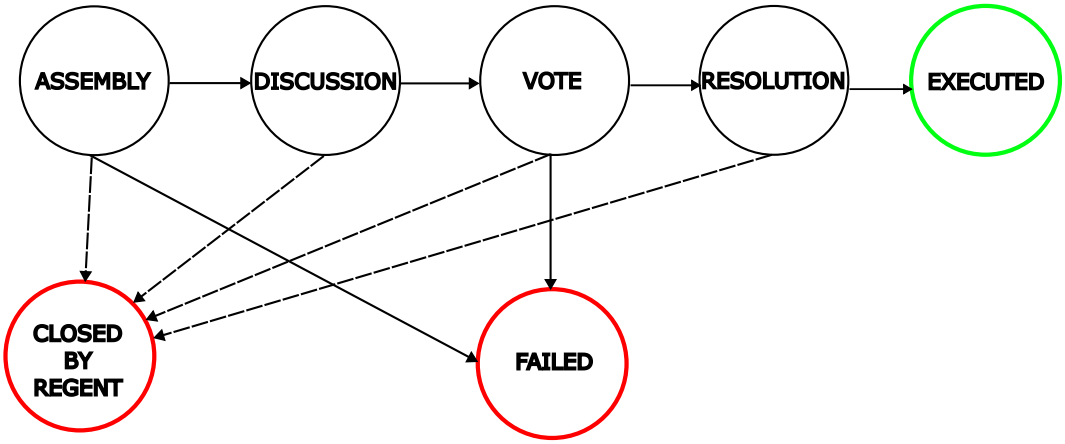

Figure 2.2 Flow of the deal phases.

2.6 Resources

The primary resource of our game is called **Reputation**. The amount of reputation a player has is the primary win condition of the game. The details are expanded upon in future sections.

We implemented some additional resources that are needed to execute stratagems. At the start of each turn, all players receive these resources, based on default gains and the tiles they own. The amount of resources a player has limits which stratagems they can use. This limitation encourages players to engage in diplomacy to trade for resources they need to use their stratagems.

From this perspective, the particulars of which resources are implemented do not matter.

2.7 Map

In addition to the numerical resources, players can also take control of sections of the game world's map. Owning these can be considered to be a resource. For the purposes of diplomacy, map presentation is not all that relevant. From this perspective, only a few abstract functionalities are required. The map can be separated into sections that can be owned by players. Any additional functionality and purpose is determined by the actions and the associated game mechanics. For example, map sections can generate additional resources.

In our implementation we choose to represent map as square grid of tiles. Each tile and its resources can be utilized by stratagems.

2.8 Possible design improvements

Here we outline several possible future improvements that could be made to the game:

1. **Events**. Strategy games such as Europa Universalis IV incorporate rich event systems. The player is presented with a random offer, where they have to make a choice. This can make each play-through more diverse and unique as is outlined in *My Elephant in the Room: An 'Old World' Postmortem* [15].

2. **City-Management**. Each player is given a city. They can manage the city to improve it and again benefits. Different benefits encourage different strategies. Stratagems are used to improve and manage the city. Cities can become objectives, enabling new win conditions and choices to be considered when planning a path to victory.

3. **Incomplete information**. Players can use diplomacy to trade for information. For example, the amount of resources a player has, or insight into that player's actions. Forcing players to consider credibility of information provides further interesting design decisions.

    (a) Espionage allows players to gather information.

    (b) Subterfuge allows fabricating false information.

4. **Congress.** Players can vote on global changes to the nation. For example, transforming the country to non-authoritative form of government. Similar to implementations present in Stellaris (Galactic Community) or Civilization VI (The World Congress).

5. **Extending deals.** The Third phase, *Definition of additional discussion points.*, can be extended to improve the deal process by introducing different types of points.

   (a) Supplementary points as reactions to the primary points. Point suggesting alliance between two players, can be supplemented to also include guarantees on third players.

   (b) Appeasement points that would allow other players to bid counteroffers similarly to an auction.

2.9 Premise

Politics between countries is an interesting concept and provides grander scale to the game. However, once nations are involved, player expectations are drawn towards conquest. This jeopardizes diplomacy as the main focus of our game.

For these reasons, we choose to focus on small-scale politics internal to single country. The player's ultimate goal is to seize the throne, as the country starts the game without a ruler. This gives players a clear goal from start to finish.

The default way to seize the throne is to achieve the highest reputation. Reputation is obtained by taking part in successful deals based on the amount of points and the participant count. This win condition alone would make players lacking in reputation lose interest as their chances to win are diminished. In order to amend this there is a stratagem that lets its user usurp an empty throne. If the throne is occupied, players can negotiate to make the ruler abdicate or resort to violence and remove him forcefully. Usurping the throne, however, involves a gamble as there is no certainty of the ability of your adversaries to remove you from power. Finally, an opponent with the right stratagem can offer to support the usurper's right to the throne. The usurper can accept or reject this offer. This serves as a way to avoid complete defeat as the supporters are listed as partial winners, if their chosen usurper wins. Such option mitigates the frustration of losing.

Chapter 3

Developer Documentation

We recommend becoming familiar with the game before delving into this chapter. The game is described in Chapter 4. Setting up development environment is covered in Section 3.2.

3.1 Development environment and tools

The game is being developed in *Unreal Engine 5* [26]. The primary motivation for choosing it over alternatives such as *Unity* [27], *Godot* [28] and others, is its proper built-in multiplayer support. Compared to alternatives, Unreal Engine has an out-of-the-box built-in multiplayer, making it easier to prototype and develop multiplayer games. When considering other aspects such as familiarity, documentation, and overall productivity of tools, Unreal Engine might not rank as the best choice individually. However, as a comprehensive package, it stands out as the best choice.

Among other tools used primarily for temporary asset creation were free or open-source tools, such as *Krita* [29] and *Blender* [30].

3.2 Setting up development environment

Setting up the development environment from source requires *Unreal Engine 5.2.1* and *Visual Studio*. The setup can be done by following these steps:

1. Download Unreal Engine 5.2.1 following the instructions at:
   `https://docs.unrealengine.com/5.2/en-US/installing-unreal-engine/`

2. Download Visual Studio following the instructions at:
   `https://docs.unrealengine.com/5.2/en-US/setting-up-visual-studio-development-environment-for-cplusplus-projects-in-unreal-engine/`
3. Clone repository from:
   `https://github.com/Woprok/UnrealDiplomacy`
4. Go to `../Sources/UnrealDiplomacy`.
5. Right click on `UnrealDiplomacy.uproject` and select `Generate Visual Studio-project files`.
6. Open `UnrealDiplomacy.sln` using Visual Studio.
7. Choose either `DebugGame Editor` or `Development Editor` build configuration and compile and run the solution.
8. You can now run the game through play in editor:
   https://docs.unrealengine.com/5.2/en-US/in-editor-testing-play-and-simulate-in-unreal-engine/
9. Alternatively you can follow instructions to package the project and then running it: `https://docs.unrealengine.com/5.2/en-US/packaging-unreal-engine-projects/`

3.3 Project Structure

The project is primarily written in C++ following the Unreal Engine conventions. The main use for Unreal Engine Blueprints is for user interfaces (UI) and instancing types that require further configuration. Blueprints types explicitly mentioned in this chapter contain *BP* prefix, such as *BP_UDSkirmishHUD*.

The project uses the following file structure:

- **Core** - base types for UI, game logic and world simulation.
- **Menu** - types used by *Menu map*.
- **Lobby** - defines UI used in lobby.
- **Skirmish** - types used by *Skirmish map* and blueprint instances for resources, tiles, materials and more.

3.4 Architecture and Networking

Unreal Concepts

1. Game Mode - handles game rules and players
2. Game State - object for sharing state of the game, owned by server and replicated on clients
3. Player Controller - represents client connected to server

The default for Unreal Engine is to operate as client-server application [31]. Consequently, one of the main points we had to define was responsibilities for both client and server. The server side holds all authority and verifies that actions done by clients are valid. Thus clients responsibilities are limited to presenting local state of the world that is updated only by server approved actions.

GameMode provided by *Unreal Engine* is available only on the server and in general it's responsible for handling players and game rules for a game match [32]. This led us to define boundaries between client and server as shown in Figure 3.1. Each client owns single *PlayerController* [33] and server owns both *GameMode* and *GameState*. *PlayerController* is responsible for communicating player actions with server through the *GameState* and managing its local world state, based on actions received by server. The *GameMode* receives actions from the *GameState* that are processed and then send back to players.

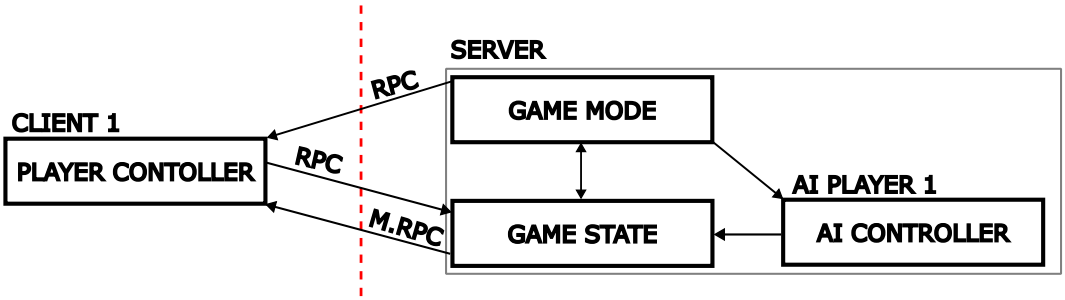

Figure 3.1 Split between the client and the server. The client is represented by PlayerController, AI agents are a part of the server, each represented by a single AIController.

3.4.1 Synchronization

Unreal Engine offers multiple ways to synchronize Client and Server state of the world. For our project we choose to utilize as main method of synchronization Remote Procedure Calls (RPCs) [34]. Alternatives in form of actor and property

replication are not reliable for deterministic simulation that is required to keep all clients and server synchronized.

Client and server communicate changes to the state of the game by exchanging either **FUDCommandData** or **FUDActionData**. *FUDCommandData* is used to communicate non game state related changes: start of the game, host closing the game and player leaving the game. *FUDActionData* on the other hand communicates all changes that are requested by client or needs to be applied to *UUDWorldState* once verified by server. Finally, *FUDActionArray* is used for initial synchronization of the *UUDWorldState* between newly connected client and the server.

Table 3.1 lists all RPCs used for synchronizing client and server. *AUDSkirmishPlayerController*, *AUDSkirmishGameState* and *AUDSkirmishGameMode* are our implementaions of *PlayerController*, *GameState* and *GameMode*. Client invokes RPCs on *PlayerController* and these are processed by *GameState*. The *GameMode* on the server can invoke RPC directly on the *PlayerController* or indirectly by invoking a multicast on the *GameState* that is processed on local instance of *GameState* on the client.

RPC function	**Caller**
UFUNCTION(Client, Reliable) void ClientcastReceiveActionFromServer (FUDActionData serverData)	Server GameMode
UFUNCTION(BlueprintCallable, Server, Reliable) void ServercastSendActionToServer (FUDActionData clientData)	Client PlayerController
UFUNCTION(Client, Reliable) void ClientcastInitialSyncReceiveFromServer (FUDActionArray actions)	Server GameMode
UFUNCTION(Server, Reliable) void ServercastInitialSyncRequestToServer (int32 controllerId, int32 firstKnown)	Client PlayerController
UFUNCTION(BlueprintCallable, Server, Reliable) void ServercastSendCommandToServer (FUDCommandData clientData)	Client PlayerController
UFUNCTION(NetMulticast, Reliable) void MulticastSendActionToAllClients (FUDActionData serverData)	Server GameState

Table 3.1 Table of RPCs

3.5 Simulation

The central part of the game is simulation of the world. All actions done by players and by the world are applied to the world in form of actions. Each action is represented by a simple struct **FUDActionData** that holds all necessary information for an action to be executed. Actions requested by players or AI agents are received by *AUDSkirmishGameMode* and passed to *AUDWorldSimulation* for execution. *AUDWorldSimulation* verifies execution of the action over the server world state. Verifying the action on the server prevents players from potentially cheating. Once the action is verified and executed on the server it is sent to all clients. *AUDSkirmishPlayerController* receives the action and passes it to local *AUDWorldSimulation* to be directly executed without any verification. Finally, local simulation propagates update notification on the client so changes can be shown to the user. This process is showcased in Figure 3.2. The server maintains a separate copy of a *UUDWorldState* for each player, but only the server world state is checked for validity.

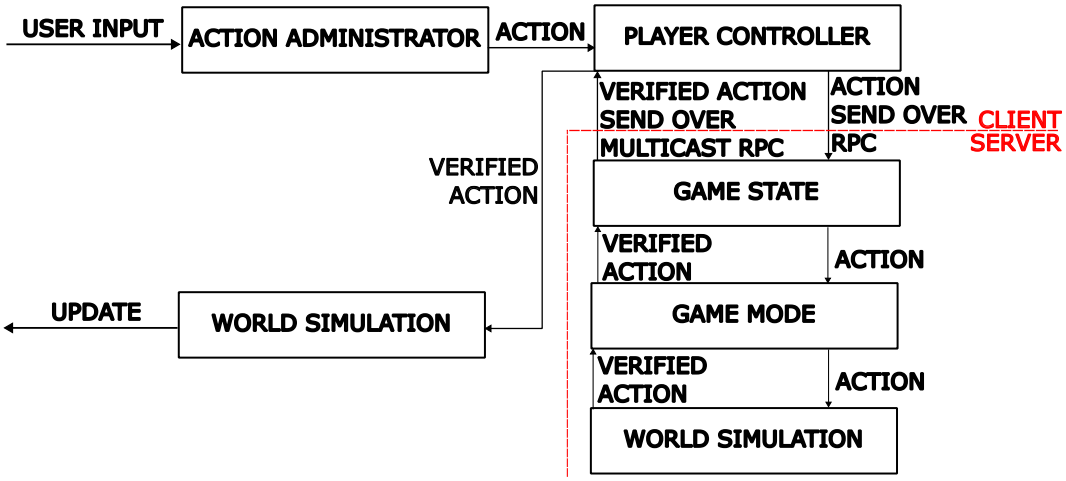

Figure 3.2 The processing of an action invoked by the client.

The process of verification and execution of the action is done in multiple steps by processing the *FUDActionData*. The process can be summarized into following steps:

1. Action responsible for execution of *FUDActionData* is retrieved from **UUDActionManager**.

2. Action checks if provided *FUDActionData* can be applied to server owned **UUDWorldState**. If the check fails, execution halts.

3. **UUDStratagemOperationManager** checks for additional restrictions and requirements that are applied to stratagems, such as execution limit and cost. If these requirements are met, additional actions are executed. Otherwise, the execution is halted. Actions that are not stratagems are skipped.

4. *FUDActionData* are assigned new UniqueActionId.

5. *FUDActionData* are stored in execution history.

6. Action is executed on all available world states.

7. *FUDActionData* are send to all clients and AI agents

8. Action checks, if it requires specific continuations, retrieves them from the action and executes them.

9. **AUDWorldArbiter** verifies, if the action triggered end of the game and determines the winner and ends the game.

The process for verification and execution of the action can trigger execution of multiple actions. All checks are repeated for subsequent actions before attempting their execution. Thus single actions, such as *UUDDealActionContractExecute* triggers execution of multiple actions. This solves the problem with execution of larger changes to the *UUDWorldState* as all that is required for an action is to be able to create continuations, that achieve desired outcome. For example *UUDDealActionContractExecute* triggers execution of all points defined in the deal by executing their resolutions, that are already present as part of *UUDDealState* as *FUDActionData*. Thus a complex customization that is required in deals is achieved by creating new actions and updating their parameters. Then they are converted to a *FUDActionData* that can be executed as any other action.

The details concerning *UUDActionManager* and actions are expanded upon in future subsection.

UUDStratagemOperationManager has two roles. First is to verify that stratagems are executed only once per turn for each player. Second is to apply resource cost to actions that are not free.

UUDWorldArbiter is used to ensure that game ends and always has a winner. By default it applies victory to player with highest total reputation.

3.5.1 World State

Current state of the world is represented by *UUDWorldState*. The most important parts of the *UUDWorldState* are represented by *UUDMapState*, collection of *UUDFactionState* and collection of *UUDDealState*. Actions registered with *UUDActionManager* update the *UUDWorldState* by their *Execute* method. Factions are represented as *UUDFactionState* and controlled either by player or AI agent. Gaia faction is unique faction that has *unique id set to 0* and is controlled by AI agent of type *AUDSkirmishGaiaAIController*. *UUDDealState* represents single deal created by players. *UUDMapState* is wrapper over individual tiles of type *UUDTileState* and map specific details such as seed and size.

3.5.2 Actions, Modifiers and Resources

UUDActionManager is responsible for providing actions to *UUDWorldSimulation* as well as filtering and access to details about actions. *IUDActionInterface* defines common interface used by all actions. We differentiate between following action types based on their purpose:

- *System* - creation of the world state and turn management.
- *Gaia* - world updates defined by game rules.
- *Game* - stratagems and related actions.
- *Deal* - creation, updates and management of deals.
- *Decision* - creation of interactions, offers, requests and demands and their resolution.
- *Setting* - updates to settings in lobby for both world and player.

UUDModifierManager handles all modifiers from creation on *UUDFactionState* and *UUDModifierState* to their eventual removal. *IUDModifierInterface* is used to define two types of modifiers:

- *UUDTileModifiers* - associated with *UUDTileState*.
- *UUDFactionModifiers* - associated with *UUDFactionState*

UUDResourceManager manages all resources and common operations with the resources over the *UUDFactionState*. Resources are defined as types implementing *IUDResourceInterface*.

3.5.3 Action Administrator

UUDActionAdministrator main purpose is to act as a model for viewmodels and map. Thus it provides extensive list of functions for transforming raw *UUDWorldState* to more specialized structs defined in `UDModelStructs.h`.

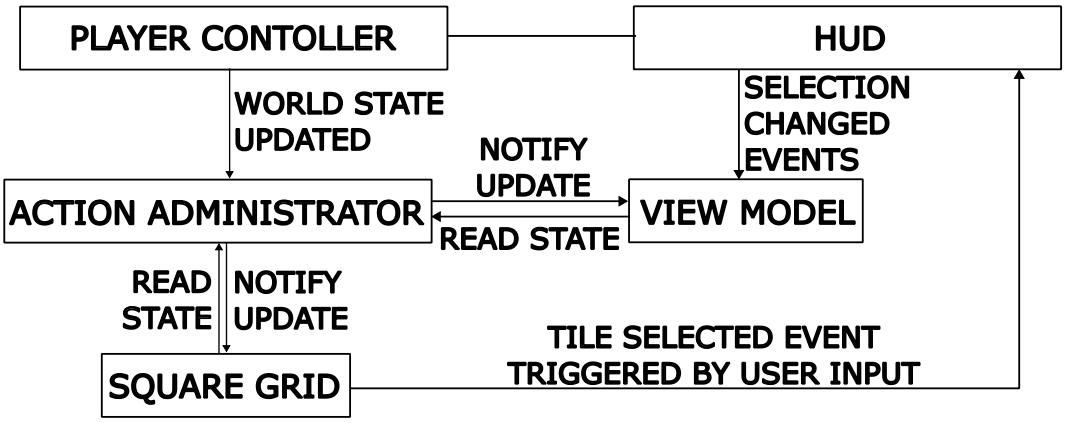

Figure 3.3 The overview of the UUDActionAdministrator role in the updates on the client.

3.6 Important Types

3.6.1 User Interfaces

User Interfaces (UI) are written in Model-View-ViewModel pattern using *UMG ViewModel Plugin* [35].

The entry point for UI is either *BP_UDMenuHUD* and *BP_UDSkirmishHUD* both derived from common ancestor *AUDHUD*. The UI is defined by *screens* that are composed of one or more widgets. Each widget has a viewmodel that is associated with it. For example *LobbyScreen* that is part of *BP_UDSkirmishHUD* is defined as combination of:

- Lobby - *WBP_LobbyView* and *UUDLobbyViewModel*
- LobbyBackground - *WBP_BackgroundView* and *UUDBackgroundViewModel*

AUDHUD is responsible for creating, displaying and switching between screens. The individual views and viewmodels are created and managed by *UUDUserWidgetManager* and *UUDViewModelManager* both are part of *AUDHUD*. *UUDSkirmishHUD* provides *UUDViewModelManager* with a reference to the *UUDActionAdministrator* that is used as the model by skirmish viewmodels. In

addition, *AUDHUD* exposes API for creating new viewmodels for nested widgets.

UUDViewModel is ancestor for all viewmodels used by the game. *UMG ViewModel Plugin* does not have sufficiently rich options for binding all fields on widgets. Thus binding *FText* is done by function specialization. Nested viewmodels are bound to their widgets by utilizing *FUDViewModelList* and *FUDViewModelContent* that wrap the underlying viewmodel and bind it to a custom property *UUDListView* and *UUDContentUserWidget*.

Views are designed in blueprints. Each view has a C++ ancestor that defines viewmodel binding and subscribes delegates, such as button click to a specific viewmodel function. We distinguish four types of UserWidgets based on their functionality and purpose:

1. *UUDUserWidget* - basic functionality required by all widgets, used by static screen views.
2. *UUDListEntryUserWidget* - represents single item that will be part of list, used by item views.
3. *UUDContentUserWidget* - enables binding viewmodel to nested widgets, used by views that are part of another view.
4. *UUDWindowUserWidget* - movement functionality for various windows, used by movable screen views.

The visualization of the UI and its dependencies can be observed in Figure 3.4.

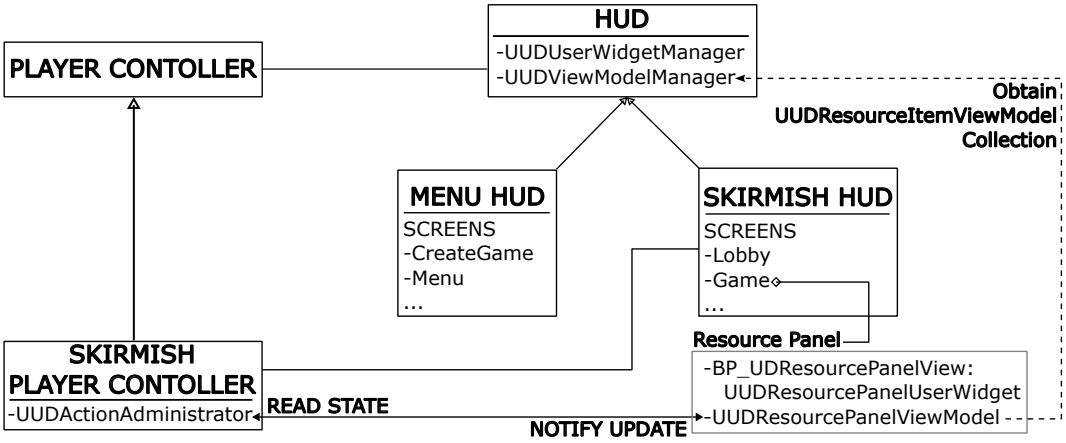

Figure 3.4 Structure of the UI dependencies for Resource Panel that is part of Game Screen.

3.6.2 Game Instance

Each client has singleton of *UUDGameInstance* for the runtime of a game [36]. *UUDGameInstance* is used for instancing all important stateless managers: *UUDActionManager*, *UUDModifierManager* and *UUDResourceManager*.
The following subsystems are part of *UUDGameInstance*:

- *UUDSessionSubsystem* covers session functionality provided by online subsystems [37]. Our current implementation uses default *OnlineSubsystemNull*.
- *UUDExceptionManagerSubsystem* tracks network errors and stores most recent error that needs to be displayed to user.

3.6.3 Maps

The project has two main maps *Menu* and *Skirmish*.

Menu is default starting map for the game. *BP_UDMenuGameMode* is set as a game mode for the *Menu* map. This map is used for all non-game sections of the game.

Skirmish is map used for playing the game. It loads when player enters the lobby and remains active throughout the entire duration of the game match. After game ends, players are returned to the *Menu* map. *Skirmish* uses *BP_UDSkirmishGameMode* as a game mode.

3.6.4 Controls and Camera

AUDPlayerController defines input mode as both game and UI. Camera controls are defined as part of invisible *AUDSkirmishPawn* controlled by player using the *Enhanced Input Plugin* [38] for managing input.

3.6.5 World Map

The world map consists of tiles. Each represented as *UDSquareTile* and dynamically created by *AUDSquareGrid* based on current *UUDWorldState*. The *UUDMapState* is generated by *UUDWorldGenerator* and factions are distributed on the generated world map by *UUDWorldFactionGenerator*. Blueprints are used to define individual tile types and configure additional details, such as resource to tile type binding. These are part of the *BP_UDSquareGrid*.

3.7 Expanding Actions, Modifiers and Resources

The project was aimed to be built as extensible as possible. Thus adding most common types: Actions, Modifiers or Resources can be done by creating new instance inheriting from their respective interfaces: *IUDActionInterface*, *IUDModifierInterface* or *IUDResourceInterface*. Once new type is created, it needs to be registered to *UUDActionDatabase*, *UUDModifierDatabase* or *UUDResourceDatabase* respectively.

Action cost is defined in table that's part of *BP_UDStratagemOperationManager*.

Resources can be spawned on the map. Each resource that can be generated on the map, needs new instance of *BP_AUDSquareTile*, that is added to *BP_AUDSquareGrid*. Finally resources that have blueprint instance can be add to *BP_UDResourceManager*.

3.8 Working with AI API

All AI agents that are part of the game are either *AUDSkirmishAIController* for faction agents or *AUDSkirmishGaiaAIController* for server world agent. Both are derived from *AUDAIController*. *AUDAIController* defines interface required by *AUDSkirmishGameMode* to be able to communicate with AI agents.

AUDAIController has access to *UUDActionAdministrator*. *UUDActionAdministrator* utilizes types defined in *UDModelStructs.h* to provide variety of functions that make *UUDWorldState* easier to work with. Current options provided by *UUDActionAdministrator* might not be sufficient for more complex agents. In that case, the world state can be accessed directly through method *UUDWorldState* GetOverseeingState()* provided by *UUDActionAdministrator*.

AUDAIController exposes following overridable virtual methods:

- *void ProcessPlayableAction(const FUDActionData& executedAction)* - receives all action from the lobby until the end of game.
- *void ProcessPreGamePlay()* - enables selection of the stratagems, invoked only before the game starts.
- *void ProcessInTurnPlay()* - called at the start of a new turn and invoked by received actions until the agent uses *UUDSystemActionTurnFinish*.

- *void ProcessOutTurnPlay()* - invoked by received actions after the agent used *UUDSystemActionTurnFinish* in the current turn. The agent can perform any action, but it's no longer able to delay the end of the current turn.
- *void ProcessIntermezzoPlay()* - reserved for the world agent to manage transition between the turns.

These methods simplify the actions received by *void OnActionExecuted(FUDActionData executedAction)* into more manageable parts, based on the current state of the world.

The current API is limited and unsuitable for extended calculations as it's invoked directly by *AUDSkirmishGameMode* whenever it receives actions from *AUDWorldSimulation*.

Chapter 4

User documentation

In this chapter, we will briefly go over how to run and play the game.

4.1 Obtaining and Running the Game

Game requires at least Windows 10 operating system. We advise to run it on a decent modern hardware.

1. Download latest version from:
   `https://github.com/Woprok/UnrealDiplomacy/releases`
2. Extract the game and navigate to the `Windows` folder.
3. Run `UnrealDiplomacy.exe`.

4.2 Menu and Settings

Game has menu, that enables to either create new game, join game or modify settings. Figure 4.1 shows the menu.
Settings are limited to resolution and window mode. As seen in Figure 4.2.

4.2.1 Create and Join

At the moment the game is using the default null online subsystem. Thus only local and LAN play is supported.

Create

Creating a game is shown in Figure 4.3.

1. Go to `Create Game`.
2. Optional change session name.
3. Click `New Game` to create new lobby for others to join.

Join

The easiest way to connect two devices is through the Direct Connect. One player has to create a game. The other player can join through the Direct Connect. In such a case players should have devices on the same LAN or VLAN. Default Port is 7777 assuming only one instance is running and hosting the game. Joining a game is shown in Figure 4.4.

1. Go to `Join Game`.
2. Refresh will attempt to find lobbies.
3. Either click on one of the found lobbies or connect through the Direct Connect.

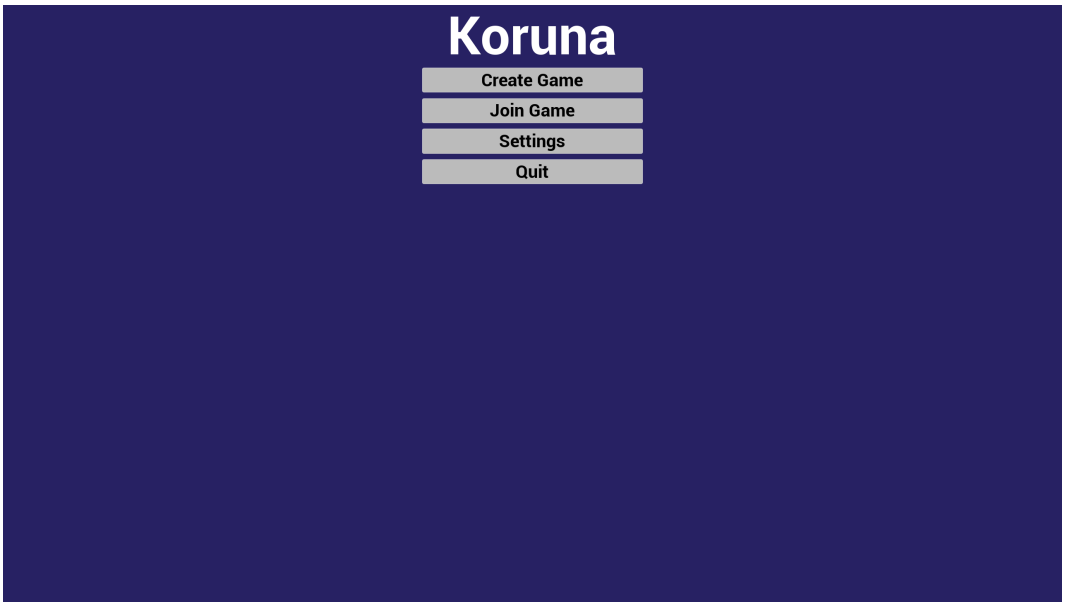

Figure 4.1 Main menu

Figure 4.2 Settings

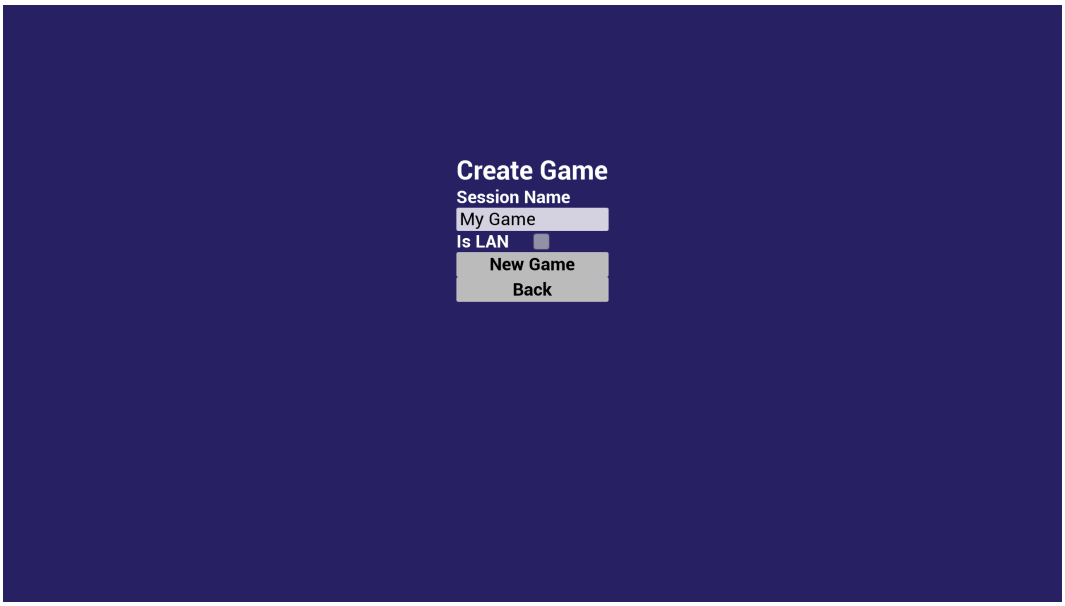

Figure 4.3 Create Game

Figure 4.4 Join Game

4.3 Lobby

Lobby as seen in Figure 4.5 enables configuration of both game and faction parameters. The host can define following game parameters.

1. **AI Count** - determines the amount of AI opponents.
2. **Map Seed** - randomness used by map generator.
3. **Map Width** and **Map Height** - amount of tiles to generate in rows and columns.
4. **Stratagems Points** - amount of points players can use to select stratagems.

All players can change:

1. **Faction Name** - their identifier visible to other players.
2. **Stratagems** - can be selected, their total point cost has to be at most the specified point cap.

Once all players are prepared, the host can start the game by clicking `Start`.

Figure 4.5 Lobby from the host perspective.

4.4 Gameplay

Once the game starts players are shown initial game screen. Initial game screen is detailed in Figure 4.6.

Resource Panel

Game has 6 resources:
Reputation [Purple] is the primary resource and having most reputation is the default condition for winning a game after the final turn.
Gold [Yellow] is used to pay for activating almost all stratagems.
Food [Green] and **Materials** [Brown] are spent on stratagems that create buildings.
Luxuries [Pink] are used on stratagems and are paid on maintenance at the start of each turn for each owned tile.
Manpower [Blue] is used on military stratagems such as Burglary, Raid and most throne related stratagems such as Usurp Throne.

Resources are obtained at the start of each turn automatically and by stratagems such as Exploit Tile or Raid Tile. Reputation is a special resource that is also rewarded for participating in deals that passed.

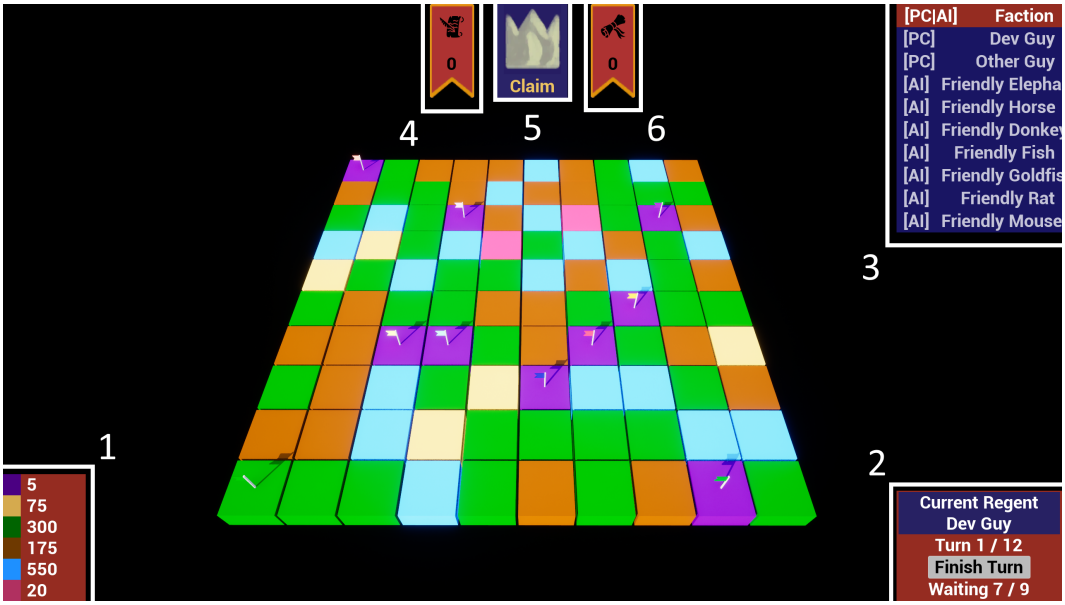

Figure 4.6 Initial game screen. In the bottom left corner (1) is a resource panel for the current player. In the bottom right corner (2) is a turn panel. In the upper right corner (3) is a faction panel. In the upper center of the screen in order from left to right deals (4), crown (5), messages (6).

Turn Panel

Top of the turn panel shows current regent. Regent is a faction that has right to create one new deal during regent's turn. Game is played in 12 turns as is shown in the panel. Finish turn button will lock you in for waiting in for next turn. All players needs to finish their turn for next turn to begin as is shown at the bottom of the panel.

Faction Panel

This panel shows all factions that are participating in the game. Clicking on a faction in the list will open a Faction Management window as seen in Figure 4.7. Here, player can perform bilateral diplomacy with the selected faction. It also shows any stratagems used by the faction this turn. Interactions will execute stratagems directly without other faction's consent. Offers, Requests and Demands require other faction's consent for execution. They have option to either accept or reject it. Interactions and Offers concern current player's stratagems, where Requests and Demands concern other player's stratagems. Finally, Demands can utilize stratagems such as Illicit Rumour or Burglary as a threat for rejecting.

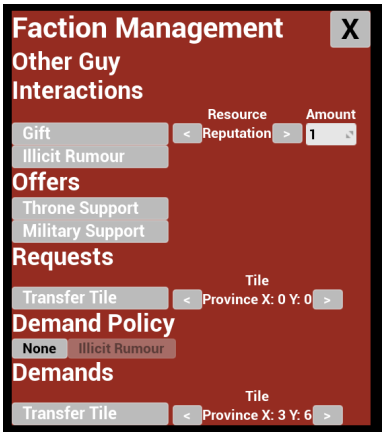

Figure 4.7 Faction Management.

Crown

Crown can be usurped by a player using the Usurp Throne stratagem. Once usurped, all players can see usurper's name under the crown as is shown in Figure 4.8b. If usurper stays on the throne until the end of the game, they will claim victory regardless of reputation. Player can abdicate voluntarily or as a part of a deal. Option to abdicate can be seen in Figure 4.8a. Contest Throne stratagem is used to overthrow the usurper and take throne for yourself. Liberate Throne stratagem will remove usurper from the throne and liberator will receive all of usurper's reputation. Both Liberate and Contest are resolved by amount of manpower on both sides of conflict. Manpower is counted as a sum of faction's manpower and all of its military supporters manpower.

Messages

Messages display amount of pending messages. By clicking on the messages will open Message Management window. This window contains Offers, Requests and Demands that you can either accept or reject. Consequences of rejecting are also presented as seen in Figure 4.9.

Tiles and Tile Management

In middle of the screen game shows current game map. Map is composed of square tiles. Each tile has a color of a resource it contains as described in the Resource Panel section. Each owned tile has a flag with a faction color. Clicking on the tile opens Tile Management window as seen in Figure 4.10. This window contains basic information about the tile, stratagems used on the tile this turn and tile specific stratagems that player can use.

(a) Usurper version of the crown. **(b)** Usurped crown as seen by others.

Figure 4.8 Crowns.

Figure 4.9 Demand from other faction in Message Management window.

Figure 4.10 Selected tile and Tile Management window.

Deals

Deals display amount of active deals in which player is participating. By clicking on the deals will open Deal Management window. Players can view either current or previous deals by switching between Active and History and then using arrows to navigate.

Current regent can create deal by clicking the Create button in header of the window. Deals progress in five phases:

1. **Assemble Phase** covers invitation and assembly of participants. See Figure 4.11.
2. **Discussion Phase** allows creation of the discussion points. See Figure 4.12.
3. **Vote Phase** bids players to express their satisfaction with the deal.
4. **Resolution Phase** bids players to decide whether they accept or abandon each point that concerns them. See Figure 4.13.
5. **Execution Phase** allows regent to execute all the decisions made by players during the resolution phase.

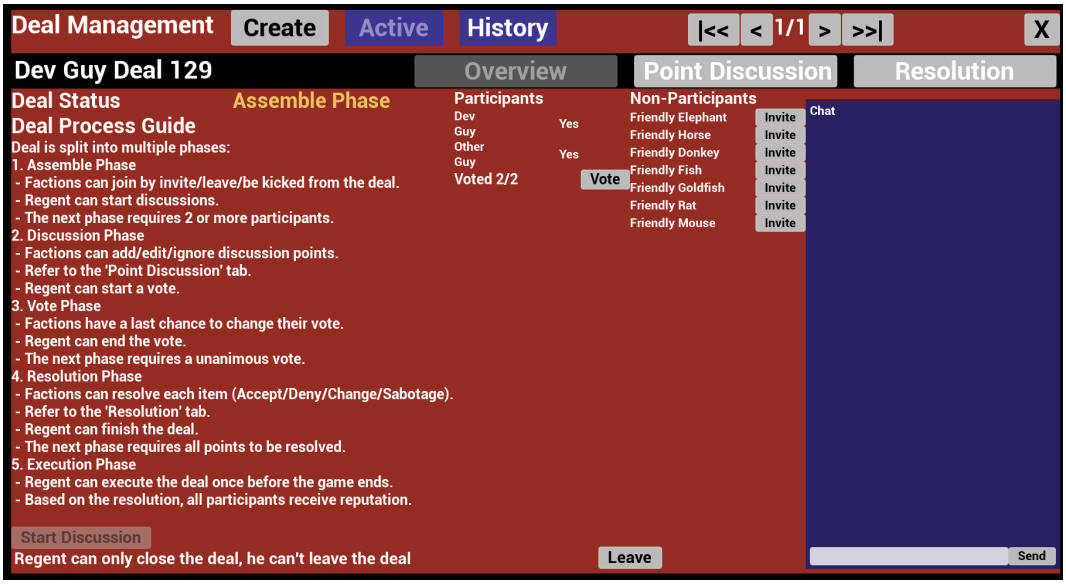

Figure 4.11 Deal Overview.

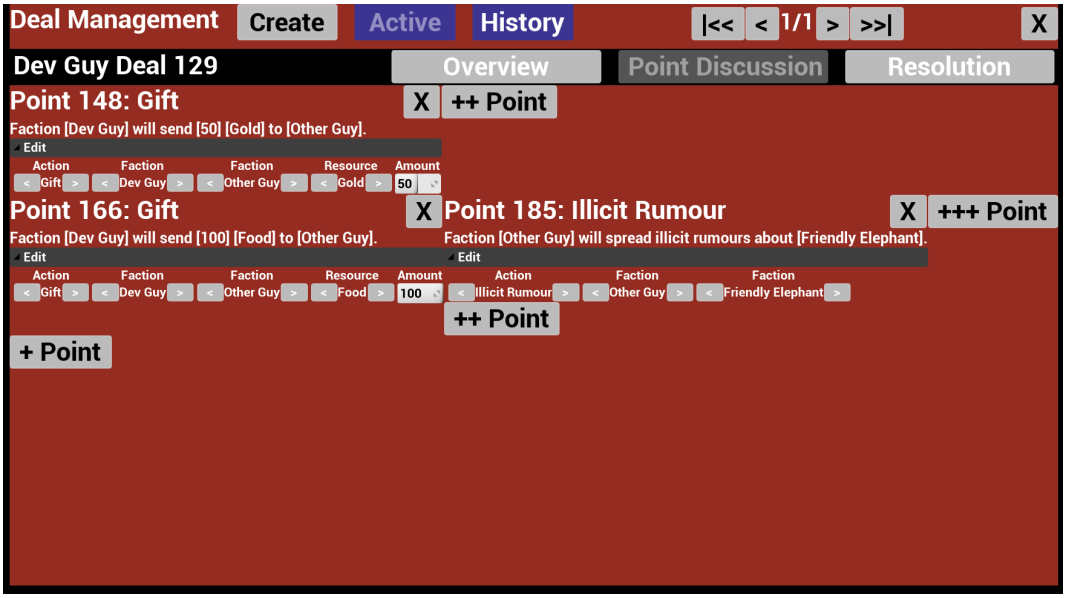

Figure 4.12 Deal Point Discussion.

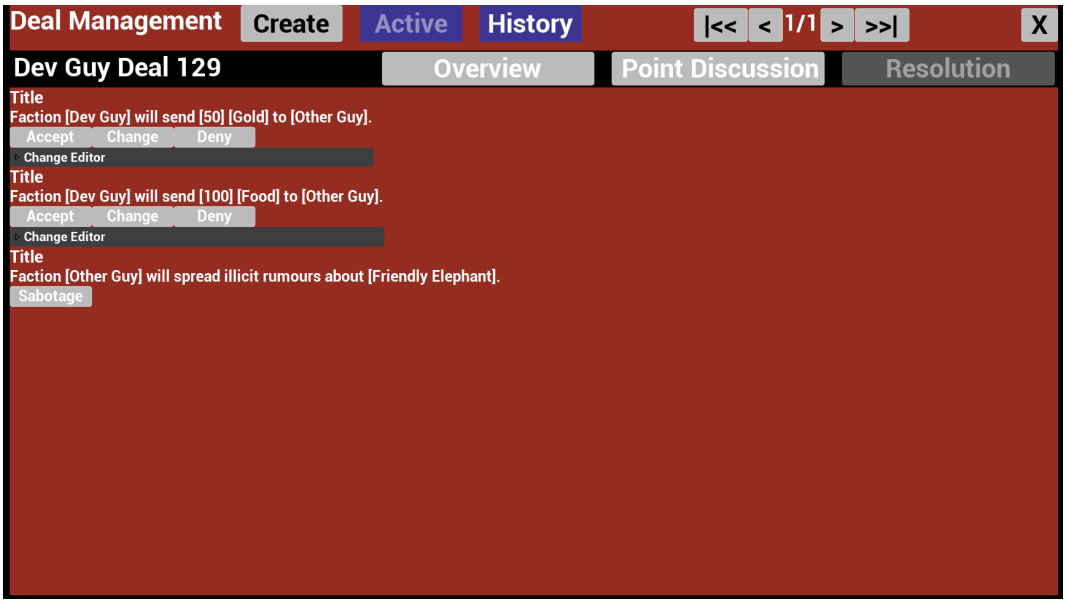

Figure 4.13 Deal Resolution.

4.4.1 Controls

A list of keybinds is presented in Table 4.1.

Key	Effect
W / Arrow Up	move camera up
S / Arrow Down	move camera down
A / Arrow Left	move camera left
D / Arrow Right	move camera right
Q	rotate left
E	rotate right
R	reset camera
Middle Mouse Button + Mouse Movement	rotate camera
Left Control + Left Shift	increment parameter by 100
Left Shift	increment parameter by 50
Left Control	increment parameter by 10

Table 4.1 Hotkeys

4.4.2 Gameplay Overview

Game can be won by two different means. The game is won by the current holder of the crown by the end of the final turn. If there is no holder, the game is won by the faction with the highest reputation. Alternatively player can avoid defeat by having in use Throne Support stratagem on winning player.

Once players are in the lobby, they can choose their stratagems. Choice of stratagems creates unique strategies and options available in the match to use as well as paths that can lead to victory. The individual stratagems can be seen in Table 4.2 and Table 4.3.

The game is played in 12 turns. The first regent is player that created the game. At the beginning of each turn the regent title is passed to next in order players joined.

During the turn players can utilize stratagems they selected before the game or obtained from other faction with Stratagem Share stratagem. Due to limited natures of stratagems, players must rely on diplomacy to outsmart opponents and achieve the victory.

It's important to remember that the regent is the only one who can create the deal and invite other players to participate. Participating is very beneficial as

deals are the main source of reputation, and they allow players to work with or against each other. In addition deals enable players to use stratagems owned by other participants. The Gift stratagem and Abdicate throne actions are always available for use in the deals.

4.4.3 AI Players

One of our goals was to create an environment for development of complex AI agents. The prototype has an example AI agent. This AI agent is capable of showcasing the basic functionality.

Before the game starts, the example AI agent will attempt to select following five stratagems in this order: Gift, Take Tile, Military Support, End Military Support and Throne Liberate.

At the start of each turn and after every action, the AI agent will resolve all pending messages. If the message is of type demand, the author of the message is marked as an enemy until the end of the game. Otherwise the message is accepted.

For any deal requiring the AI agent to vote, it decides to vote no if any enemy is included, otherwise it votes yes. The AI agent always accepts all points in the resolution phase of the deal.

Each time the AI agent receives a gift of any size and resource, it will mark the giver as a friend unless they are already an enemy.

Once per turn, before the AI agent finishes the turn it will perform following sequence of actions:

- It will end military support of the usurper or one enemy faction that already had AI agent support.
- If any usurper is present, it will attempt to liberate the throne.
- If AI agent has over 100 gold, it will give it to a friendly player. Otherwise it will grant it military support. The player is then removed from the list of friends.
- Finally, it tries to take neutral tile for itself.

Stratagem	Cost	Description
Stratagem Share	5 Gold	Allows the player to share his stratagem with the target. The target can use the stratagem independently as his own.
Gift	5 Gold	Allows the player to send his resources to another player.
Throne Usurp	20 Reputation 600 Manpower	Claims the crown for yourself. Usable only once per game.
Throne Contest	5 Gold	Fights the Usurper and if victorious, claims the crown. Thus replacing the usurper.
Throne Liberate	5 Gold	Fights the Usurper and if victorious, take all his remaining reputation for yourself. Thus removing the usurper.
Throne Support	5 Gold	Provides reputation to the target. Does not lose when the target wins.
End Throne Support	5 Gold	Removes the throne support modifier from the target.
Military Support	5 Gold	Provides manpower to the target.
End Military Support	5 Gold	Removes the military support modifier from the target.
Illicit Rumour	75 Gold	The target loses 5 Reputation.
Burglary	25 Gold 275 Manpower	Target loses 50 Gold and 10 Luxuries. Invoker gains 50 Gold and 10 Luxuries.

Table 4.2 Stratagems

Stratagem	Cost	Description
Tile Exploit Permit	5 Gold	Grants the permission to exploit the tile to the target.
Exploit Tile	5 Gold	Takes resources from the tile's stockpile.
Take Tile	5 Gold	Claims the tile as your own.
Transfer Tile	5 Gold	Transfers the ownership of the tile to the target.
Tile Raid	25 Gold 275 Manpower	Devastates the tile's stockpile by at least 200 and takes some resources for yourself.
Tile Build Farm	50 Gold 500 Materials	A Food Tile gains 525 Food in its stockpile, other tiles gains 25 resource to their stockpile.
Tile Build Quarry	50 Gold 500 Materials	A Material Tile gains 525 Materials in its stockpile, other tiles gains 25 resource to their stockpile.
Tile Build Manufactury	100 Gold 500 Food 500 Materials	A Luxury Tile gains 250 Luxuries in its stockpile, other tiles gains 25 resources to their stockpile.
Tile Build Palace	450 Gold 100 Luxuries	A Reputation Tile gains 100 Reputation in its stockpile.
Tile Build Fortress	100 Gold 500 Food 500 Materials	A Manpower Tile gains 550 Manpower in its stockpile, other tiles gain 25 resources to their stockpile. In addition the tile can no longer be raided.
Tile Build Trade Guild	500 Food 500 Materials	A Gold Tile gains 250 Gold in its stockpile, other tiles gain 25 resources to their stockpile.

Table 4.3 Tile Stratagems

Chapter 5

Results

We have developed a prototype that follows the design described in the Game Design chapter. Final game developed as part of this thesis demonstrates diplomacy focused gameplay, including multilateral deals.

5.1 Project Complexity

Complexity stems from both design and implementation. Diplomacy as a core of the game requires different approach than the one used commonly by strategy video games. The varied constraints of our desired game required us to come up with creative solutions such as the stratagem system. On the implementation side, our largest issue was the overall complexity of developing a multiplayer game that requires large quantity of user interfaces and options. This was compounded by the difficulty of having to learn an unfamiliar set of tools.

5.2 Artificial Intelligence

One of our goals was to create a suitable API for AI agents. We have achieved this goal by providing a rudimentary extensible AI agent. However, we did not attempt to create sophisticated AI agents and the API required by them would require further extensions.

Development of capable AI agents is important considering the future continuation of this project. Multilateral deals are already a complex problem for AI agents to handle. Very often AI is not good at matching the behavior expected by players even in bilateral environments. Often they are extremely stingy and naive at the same time. Mixing multilateral diplomacy and concepts of trust and

betrayals can allow us to explore how to balance healthy nativity, stinginess and willingness to betray others to seize the momentum.

5.3 Future Extensions

The nature of strategy games in general is that there is always a room for the introduction of new systems and refining existing ones. The same can be applied to our project.
Features worth considering have already been mentioned the section Possible design improvements of the Game Design chapter. The ones that we consider important are:

- Improved workflow for deals, to improve clarity and to speed up the process.
- Incomplete information and extensive espionage and subterfuge mechanics.
- Standard multiplayer features provided in modern competitive games: reconnect, observing, replay, saves, etc.
- Single player focused experience to introduce players to concepts as well as to provide alternative to online play.
- Extending core gameplay features and mechanics: world generation, resources and paths to victory.
- Visual overhaul and properly defining the theme and the setting of the game.
- Introducing new mechanics such as warfare, city management and many more while keeping the diplomacy focus in mind.

5.3.1 Importance of Incomplete Information and Espionage

Incomplete information is the one thing we consider absolutely essential to achieve reasonably good experience in playing this type of a game. Even in the current very limited form the game shows some problems. Currently the players have access to almost all information about the world and the opponents. Therefore, they can make decisions very accurately and almost never get surprised by the actions of other players. This heavily reduces the options that the players can utilize to achieve victory and makes it hard to manipulate deals. If we provided players with very limited information and tools to obtain new information, creating new (possibly false) information and spreading it would provide additional layer of intrigue and social interaction. Thus introducing information warfare and associated options would enrich the gameplay experience.

5.3.2 Stratagems

Stratagems are the addition which we consider to have contributed the most positively to our game design. It could arguably also be considered the most innovative approach compared to other games in the genre.

Commonly strategy games tend to lean toward either providing all options without any real limits or just limiting the amount of actions a player can take.
Stratagems, on the other hand, are a fresh approach to this concept. Instead of having access to a full toolkit, players start with only a very small portion. Ideally they want to pick stratagems they consider fundamental for their strategy, but also take options that help them engage in diplomacy with others.

One of the common problems with the standard approach is that many actions start feeling inconsequential and like chores. Stratagems are allowed to be powerful due to each player having access only to a few. Combination of this with the aim for a small amount of turns can serve as an interesting concept worth iterating and expanding upon.

Finally, stratagems are closely related to the concepts of card games and deck building. These are very close to the modern audience, providing a clear path to introducing new players to the game.

5.4 Problems of Social Interaction

Multiplayer games are well known for having never ending problem with toxicity. We can consider this one of the major hurdles needing to be properly addressed in all aspects of our game design. Negative player behavior such as intentionally disrupting the game or improper social behavior can sour the experience for all other players in the game.

Trying to fight the problem might result in creating a conflict with the original design. A common approach to reducing toxicity is to reduce the interaction surface between players. However, this goes directly against the core aspects of our design. On the other hand, reducing the importance of victory and focusing on not losing as the main goal might reduce the amount of problematic player behavior.

Thus creating paths to victory as well as paths to not losing might be one of the most important aspects to consider in overall game balance.

We will end this on a simple note: even losing should be fun. It should be the moment where everyone accepts that they were truly outplayed and appreciates the intrigues that led them there.

Conclusion

We have developed a prototype that serves as the proof of concept for a diplomacy-based multiplayer strategy game.

1. Our prototype is a working client-server game that allows player to host and join game sessions.

2. Our aim was for diplomacy to be the core mechanic. Once we realized the design within our prototype, it is clear that diplomacy really is an important aspect of the game and required to reach victory.

3. The game provides the player with a wealth of options, including bilateral diplomacy in the form of interactions, offers, requests and demands. Multilateral diplomacy is led by the deal mechanic that serves as the main way of gaining reputation.

4. Multilateral interactions are built to provide more options and to overshadow bilateral interactions in how impactful they are on the final result of the game.

5. Elements of espionage and deceptions are not included in any mechanical way. But by the nature of multilateral deals, players can opt to act dishonestly.

6. The current prototype is easy to extend with more Actions, Modifiers and Resources. This allows for a decent level of possibilities to include more gameplay options.

7. The AI API exists in a very basic form. AI is allowed to interact with the game through the same system as a player. However, we did not include an ability to provide external AI implementations in the prototype.

The scope of this project is just too extensive for a single person. To reach a better result it would need either a lot more time or a small team. The inexperience

with designing a more complex game played a role as well.

We were able to successfully create a prototype that explores the possibility of strategy games focused on diplomacy over other aspects of play.

Our design approach shows potential in exploring this area. Strategy games that put much larger impact on diplomacy as a way of achieving goals can be interesting to explore. In our case, using stratagems as the core mechanic helps the players focus on overall strategy and encourages them to use diplomacy. Deals work well in combination with the stratagems as a way to provide the options that players lack compared to standard strategy games.

In the end, this project was derived from a different idea — multilateral peace deals. Eventually we have reached the goal of making a game with multilateral deals that focuses on diplomacy. It might be close to impossible to provide an elegant solution to make multilateral deals interesting, dynamic and interactive. The nature of complex processes that incorporate multiple people will often lead to slow gameplay that ends with disagreement and is not an engaging and fun experience.

For a future project we consider two aspects especially worth investigating:

1. AI capable of using multilateral deals in the context of diplomacy-based strategy games.
2. Use of stratagems in a classic strategy game design space.

Abbreviations

RTS - Real-Time Strategy
TBS - Turn-Based Strategy
4X - Explore, Expand, Exploit and Exterminate
APM - Actions Per Minute
1v1 - A gamemode where two players face each other.
Team Games - match between two groups of players of equal size
FFA - Free For All, each player against all other players
Tall Play - strategy focused on developing small patch of land
Wide Play - strategy focused on expansion of your land
GSG - Grand Strategy Games
HRE - Holy Roman Empire

Bibliography

[1] Firaxis Games. *Sid Meier's Civilization® VI*. Oct. 2016. URL: https://civilization.com/.

[2] Paradox Development Studio. *Europa Universalis IV*. Aug. 2013. URL: https://www.paradoxinteractive.com/games/europa-universalis-iv/about.

[3] Paradox Development Studio. *Stellaris*. May 2016. URL: https://www.paradoxinteractive.com/games/stellaris/about.

[4] G. Berridge and L. Lloyd. *The Palgrave Macmillan Dictionary of Diplomacy*. Palgrave Macmillan UK, 2012. ISBN: 9781137017611. URL: https://books.google.cz/books?id=jvarq4iy5MoC.

[5] Loban Rhett. "Digitising Diplomacy: Grand Strategy Video Games as an Introductory Tool for Learning Diplomacy and International Relations". In: *DiGRA Conference - Proceedings of the 2017 DiGRA International Conference*. Melbourne, Australia: Digital Games Research Association, 2017. ISBN: ISSN 2342-9666. URL: http://www.digra.org/wp-content/uploads/digital-library/19_DIGRA2017_FP_Loban_Digitising_Diplomacy.pdf.

[6] Relic Entertainment and World's Edge. *Age of Empires IV*. Oct. 2021. URL: https://www.ageofempires.com/games/age-of-empires-iv/.

[7] Blizzard Entertainment. *StarCraft II*. July 2010. URL: https://www.ageofempires.com/games/age-of-empires-iv/.

[8] Blizzard Entertainment. *Warcraft III: Reforged*. Jan. 2020. URL: https://warcraft3.blizzard.com/en-us/.

[9] EA Los Angeles. *The Lord of the Rings: The Battle for Middle-earth*. Dec. 2004. URL: https://en.wikipedia.org/wiki/The_Lord_of_the_Rings:_The_Battle_for_Middle-earth.

[10] Miroslav Valach. *Advanced Game Settings*. Version 4.1.0.0. May 2023. URL: https://github.com/Woprok/AOE4-AdvancedGameSettings.

[11] Mohawk Games. *Old World*. May 2022. URL: https://mohawkgames.com/oldworld/.

[12] AMPLITUDE Studios. *HUMANKIND™*. Aug. 2021. URL: https://humankind.game/en.

[13] Triumph Studios. *Age of Wonders IV*. May 2023. URL: https://www.paradoxinteractive.com/games/age-of-wonders-4/about.

[14] DotEmu. *Heroes® of Might and Magic® III HD*.

[15] Soren Johnson. *My Elephant in the Room: An 'Old World' Postmortem*. Accessed: 2023-07-12. 2022. URL: https://gdcvault.com/play/1028038/My-Elephant-in-the-Room.

[16] Alex Jaffe. *Cursed Problems in Game Design*. Accessed: 2023-07-12. 2019. URL: https://gdcvault.com/play/1025756/Cursed-Problems-in-Game.

[17] Paradox Development Studio. *Crusader Kings III*. Sept. 2020. URL: https://www.paradoxinteractive.com/games/crusader-kings-iii/about.

[18] Paradox Development Studio. *Hearts of Iron IV*. June 2016. URL: https://www.paradoxinteractive.com/games/hearts-of-iron-iv/about.

[19] Paradox Development Studio. *Victoria 3*. Oct. 2022. URL: https://www.paradoxinteractive.com/games/victoria-3/about.

[20] Creative Assembly. *Total War: WARHAMMER III*. Feb. 2022. URL: https://warhammer3.totalwar.com/.

[21] Jacob Fryxelius and Isaac Fryxelius. *Terraforming Mars*. Accessed: 2023-12-19. 2016. URL: https://fryxgames.se/product/terraforming-mars/.

[22] Klaus Teuber. *The Settlers of Catan*. Accessed: 2023-12-19. 1995. URL: https://www.catan.com/.

[23] Allan B. Calhamer. *Diplomacy*. 1959.

[24] Second Dinner. *Marvel Snap*. Accessed: 2023-12-19. Oct. 2022. URL: https://www.marvelsnap.com/.

[25] Ben Brode. *Designing 'MARVEL SNAP'*. Accessed: 2023-07-12. 2023. URL: https://gdcvault.com/play/1029024/Designing-MARVEL-SNAP.

[26] Epic Games. *Unreal Engine 5*. Accessed: 2023-12-19. 1998. URL: https://www.unrealengine.com/en-US/unreal-engine-5.

[27] Unity Technologies. *Unity*. Accessed: 2023-12-19. Aug. 2005. URL: https://unity.com/.

[28] Ariel Manzur Juan Linietsky. *Godot*. Accessed: 2023-12-19. Jan. 2014. URL: https://godotengine.org/.

[29] Krita Foundation. *Krita*. Accessed: 2023-12-19. June 2005. URL: https://krita.org/en/.

[30] Blender Foundation. *Blender*. Accessed: 2023-12-19. Jan. 1994. URL: https://www.blender.org/.

[31] Epic Games. *Unreal Engine 5 - Networking Overview*. Accessed: 2023-12-19. URL: https://docs.unrealengine.com/5.2/en-US/networking-overview-for-unreal-engine/.

[32] Epic Games. *Unreal Engine 5 - Game Mode and Game State*. Accessed: 2023-12-19. URL: https://docs.unrealengine.com/5.2/en-US/game-mode-and-game-state-in-unreal-engine/.

[33] Epic Games. *Unreal Engine 5 - Player Controllers*. Accessed: 2023-12-19. URL: https://docs.unrealengine.com/5.2/en-US/player-controllers-in-unreal-engine/.

[34] Epic Games. *Unreal Engine 5 - RPCs*. Accessed: 2023-12-19. URL: https://docs.unrealengine.com/5.2/en-US/rpcs-in-unreal-engine/.

[35] Epic Games. *Unreal Engine 5 - UMG Viewmodel Plugin*. Accessed: 2023-12-19. URL: https://docs.unrealengine.com/5.2/en-US/umg-viewmodel.

[36] Epic Games. *Unreal Engine 5 - Game Instance*. Accessed: 2023-12-19. URL: https://docs.unrealengine.com/5.2/en-US/API/Runtime/Engine/Engine/UGameInstance/.

[37] Epic Games. *Unreal Engine 5 - Online Subsystems and Services*. Accessed: 2023-12-19. URL: https://docs.unrealengine.com/5.2/en-US/online-subsystems-and-services-in-unreal-engine/.

[38] Epic Games. *Unreal Engine 5 - Enhanced Input Plugin*. Accessed: 2023-12-19. URL: https://docs.unrealengine.com/5.2/en-US/enhanced-input-in-unreal-engine/.