**To Do List:**

1. Minimum Viable Product

* ~~Building Armies, Fleets, Projects~~
* ~~Earning Money, Resources~~
* ~~Basic Demographics~~
* Visualizable map, ~~Pathfinding~~
* ~~Moving, Fighting Armies and Fleets~~
  + ~~Need to increase maintenance with unit size!~~
  + ~~Need to be able to split off undersupplied units from a larger unitGroup. Also, think about splitting off ships based on health.~~
  + ~~Armies, Fleets should not give maintenance while training.~~
* ~~Resource Deficit Consequences~~
  + ~~Need to be able to re-enable projects.~~
  + ~~Need to make sure players can’t cheese re-enabling projects. Basically, once a project is re-enabled, it must wait a turn before it can start applying its effects.~~
  + ~~Need to make sure that modifiers are lost if a project is disabled.~~

1. Pre-Alpha

* ~~Disabling/~~Foreclosures
* Trade, Supply lines
* Bot, Passwords, Security
* Fog of War
* Game Information in Text Files

1. Alpha Release

* Territorial Supply
* Complex Demographics/Immigration
* Policies

1. Beta Release

* Institutions
* Technology

1. a

* 3/23/2020: New Year ~~and New Turn~~ with just nations and empty territories.
* 3/23/2020: Buildable Game Objects. (EDIT: Including building large armies)
* 3/24/2020: Running out of money/resources, disabling projects and armies.
* 4/16/2020: (Long-term) Supply and Resource reserves – storing a limited number of resources in province buildings, having supply connections from every province to the capital.
* 4/22/2020: (Long-term) Complex recruitable population system (sex ratio, age, etc.)

**Modules**

* **GameObjects.py:** Contains all the object types used in the game. These include physical Game Objects (GameObject, Building, Ship, Unit, Infantry, Cavalry, Artillery), Abstract concepts (Dates) and Player Data (Player Objects, Nations, Client States/Trade Companies).
  + Certain physical object data (Ship types, Infantry/Cavalry/Artillery Types) have default values as dictated by the Game Dictionary. In the savegame file, savegame admins may put in their own values (their own Game Dictionary data). For example, different types of military units, spaceships, etc.
* **Savegame File:** Contains the server ID, password, players, date, game dictionary (custom or default), and all player nations and nation data.
* **Game Dictionary:** Contains all the game’s “blueprints”: i.e. available technologies, policies, armaments, buildings etc.
* **File Handling:** Saves and loads data to and from files, using recursive methods to turn complex objects to and from dictionaries, which can be stored in JSON files.

**Game Objects**

* **Nation:** The main object in a game, serves as a proxy for the player.
* **Date:** Represents the current date in-game.
* **Game Object:** Any physical objects in the game world, i.e. buildings, Units (And armies), Ships (And fleets).
  + **Unit Group:** An army or fleet. Has a location, a number of either ships or different land divisions. Returns maintenance costs and finishes training or movement on new year.
    - **Division**: Has a number of men and specific number of armaments. Armaments have a cost, a movement speed, an attack and a defense.
    - **Ship:** ----- *Do I really need to explain this?*
  + **Building:** ----- *If under construction, will have “Being Built” status and will be unusable*
* **Technology:** ----- *Has a prerequisite(s), technology type and earliest year of discovery.*
* **Policy:** ----- *Basically just has effects as well as a cost for implementing (and requirements for repealing depending on the policy type). Child objects will be different policy types.*
  + *Tradition: Inherent to a nation, can be reformed out.*
  + *Reform: Almost like a policy progression tree. Branch off into different ideologies with different effects.*
  + *Law: Customizes a nation’s specialties.*
  + *Emergency: Can be enacted in times of war, low stability, economic recession, etc.*
* **Loan:** ---- *Will include interest rate, initial/remaining balance and strikes to foreclosure*
* **Population Group:** ----- *Will include size, literacy, social class, unrest, assimilation, language and religion*
  + *Slaves: Automatically suspends a player from Twitter*

**Game Events**

* **On New Turn:** Whenever the turn advances by any number of months.
* **On New Year:** Whenever the year changes, and how many year changes have occurred if multiple.

**Journal of Progress**

* **12/08:** I’ve finished the functions that create objects from dictionaries and vice versa. While I found online resources that helped me make basic functions, they did not allow me to perform operations with multi-level objects (objects with other objects as parameters) . Therefore I had to make heavy use of recursion to ensure every single piece of data in the object was translated correctly. I’m quite proud of the systems I’ve created.
* **12/13:** I’ve finished the buildings generation (life cycle) system, at least up until final creation. A building blueprint will be obtained from the Game Dictionary, and using the blueprint data the client nation will figure out if it can afford the project in terms of space, resource costs and technology requirements **(Add policy requirements)**. If the requirements are met, the building will be created using the building object constructor (\_\_init\_\_) method. This method has many default values, not all of which are used. This is because although the blueprint may have technology requirements as a key-index pair, this should not be turned into a parameter-value pair for the object itself since it is not needed and takes up too much space in the save file. I think I can work on the construction of army units now.
* **12/14:** I’ve done the same work that I did for 12/13, but for Armies. I’ve also begun figuring out a mapping system: My current plan is that the map will have physical provinces and abstract locations within them. For the system as a whole, provinces and their equivalents will be referred to as macro-locations and the abstract locations will be micro-locations*. Before I do that, though, I’ve got to finish the combining and splitting methods for each army, the retrieve effects methods for armies and buildings, all the fleet methods, loans and population groups.*
* **12/17:** I have begun work on the building effects. From an earlier concept, I have transformed the effects system into something I like quite a lot. When a building finishes completion, it will apply its completion effects immediately and its continuous effects once every turn. These applications come in the form of adding them to a dictionary of modifiers sorted by effect category. A project will add its effects to the dictionary by retrieving them from its corresponding game dictionary blueprint, and every turn the nation will apply each effect/modifier from the dictionary. If an object transfers ownership, then its effects are transferred as well. “Monetary’ category effects will also include the total maintenance costs and revenue from all of the nation’s projects combined. In essence, each modifier in the dictionary is the sum total of all modifiers from all projects within a nation. *I still have to program the functions that retrieve effects for a project from their game dictionary blueprint.*
* **12/18:** The effects system will work as follows: A nation will have a list of effects from projects and policies. When one of these is added/deleted from the nation, its effect will be added/deleted from the list, respectively. **Each effect will have “provincial or national”, “exponential or linear”, “Object” or “Demographic/Demographic range”, and “Effect and amount” parameters.** Every object will have an “apply effects” method. For example, demographics apply changes to statistical growth rates, game objects apply modifiers to maintenance or object-specific parameters (e.g. attack/defense modifiers and supply necessity), and nations add income. If a project is deleted/taken by another nation, then that project and its effect will be removed from the nation and whatever object it affected and, if applicable, transferred to the new owner. *I’m debating on whether a list of modifiers/effects is even needed. Perhaps “nation object”-specific modifiers can be stored within the nation like all other objects, the object effects can be stored in the game dictionary, and they only get referenced when applying or removing them.*
* **12/29/2019:** Finished Effects system. Each object type has a number of modifiers that get affected by building effects. *Need to add provincial (local) effects as well as basic population and resource mechanics in order to make a playable alpha version.*
* **1/6/2020:** Building map system. Map is a series of nodes (locations) each either habitable or terrain. N*eed to add terrain types (e.g. prairie, grassland, forest, jungle etc. or star, planet, moon, asteroid belt etc.*
* **1/7/2020:** A nation will have a number of territory within which are stored all the projects that the nation may own. If a project is removed from a territory, then its effects are transferred to the other nation, including local and nationwide effects. Each territory will belong to a region defined by the map file, and any projects in the region with local effects will affect every other territory in the region. *I need to 1) redo the onNewTurn and onNewYear methods for every nation, 2) Figure out regions and effects, 3) make the map and 4) make resource mechanics.*
* **1/11/2020:** Redid the entire GameObjects.py code. Removed all turn advancing methods. *Need to add a mechanism for turns advancing and objects being affected by such a thing. Also need to add a way for UnitGroups to have movement queues, custom game dictionary usage and \_\_str\_\_ methods for objects. Also might want to add “process” objects.*
* **1/12/2020:** Redid modifiers system so it is more ergonomic and aesthetically pleasing, created Util.combineDicts() method. Determined that relevant new turn processes include projects being built and the effects cycle (project completion effects are passed to the location, and then to the nation and then to all relevant objects within the nation for application or removal); unit groups mobilizing, moving or being built; international or intranational transactions and the accumulation of resources. *Will add resources as a game mechanic, where provinces have a maximum number of resources per turn, buildings increase the amount of those resources per turn that can be “mined” by the nation out of the maximum, and projects/unit groups/armaments cost resources to build. Also need to figure out the paths between graph vertices for the map*
* **1/13/2020:** Started coding in the effects cycle. Also began conceptualizing a system for transporting armies across fleet-traversable territories, involving the storage of embarked armies in fleets as a list. ~~Right now I’m considering moving the storage of armies from the nation object to territory objects.~~ *(Edit: This would be infeasible because armies from different nations can be in another nation’s territory in war or other circumstances.)* Things are going quite smoothly! *Need to do New Years processes and project initialization. ~~Considering moving the nation to a provinces-based system~~*
* **1/14/2020:** Finished the basic effects cycle. ***Next on list is making a project purchasing system, then doing mapping and finally doing new year processes.*** *Don’t need to add resources first, can just assume that they are implemented when calculating costs until after the above are finished.**Need to add local effects and resources.*
* **1/15/2020:** Movement Queue can wait until post-alpha. Added resources to provinces, realized I already accounted for them in calculating project costs.
* **1/16/2020:** Designed mapping automation system. Each node in a map file will be listed in a json dictionary, including both node details and the names of bordering nodes with edge weights attached. The automation system will take a text file of all nodes and automatically connect all of them.
* **1/18/2020:** Made method for a nation to buy an affordable project, subtracting costs and passing the project information to the territory it will be built in. Reorganized territories system, parent territory object has all the real methods and a territory is either claimable by nations or not. Sea and Land territories will be used to organize embarking/disembarking of armies. *Need to finish buildProject method for territories, which will either place a project that is being built in the location or will return a unit group that is in training to the nation. Need to be able to cancel or delete projects and transfer territories.*
* **1/19/2020:** Finished project building system.
* **1/20/2020:** Finished territory transfer system. *Need to apply effects to demographics .Post-alpha: Need to add new overlord effects to demographics.*
* **1/26/2020:** Started and finished the automatic mapping system over the past week. Also fixed some issues with the dictionary transforming system. Maps can now be made into dictionaries and vice versa. *Need to consider implementation of maps with the rest of the program as well as the other features.*
* **1/30/2020:** Got back to work on the program. New unit movement system is in place, realized that Util.isWithin() and potentially Util.isSufficient() are obsolete. *Need to perform more house cleaning to ensure program is internally consistent and up to date before final stretch.*
* **2/4/2020:** *Need to make methods to get project maintenance and add to NewYearEffects, which will be passed up to the nation.*
* **2/12/2020:** Finished onNewYear for territories & complete retrieval in Nation object. *Now need to apply all territorial effects to the nation & get newYear effects from UnitGroups. Should also take time to clean up code, make it far more readable. Also need to account for having too few resources. If resource amount = 0, all projects that require that resource stop functioning, and all units that require that resource get a massive debuff.*
* **2/19/2020:** When building a project, the target territory now must fulfill territorial requirements to be built.
* **2/20/2020:** Slightly modified the newArmyNames function to account for different forces within the national Unit Groups list. Added isEconomyAfloat, function used to know when to take out a loan or when resources are at a shortage.
* **2/22/2020:** Heavily redesigned UnitGroups along a more object-oriented and inheritence-based approach. All of them now have a “Composition” parameter and are built around that. Composition includes all divisions and units within those. Slightly modified building costs by putting all of them in separate dictionaries in gameDict data. Will make maintenance costs their own dictionary. Replaced getMaintenance() with getCosts() for redoing costs mechanics. Self.status is now a general indication of the operational status of a GameObject. *Need to finish checkEconomicStability().*
* **2/28/2020:** Designed a SaveGame object to handle all highest-level (international and above) ingame events.
* **2/29/2020:** Now have a “Test” Savegame that I can use to test the game systems.
* **3/4/2020:** Having not updated the journal for some days, I’ll fill in the progress here. “Test” savegame now has its own map and dictionary, bugs with map creation have been fixed, and I should be able to test several features.
* **3/7/2020:** MAJOR BUG: When creating a map automatically, the edges are the same for every single vertex. It turns out that the last vertex’s distances to each of the other vertices is the same as every one of the edges, meaning that the edges for each vertex are just the last vertex’s edges at the end.
* **3/8/2020:** MAJOR BUG FIXED! Vertex object initially had default value “{}” for edges attribute, did not work because all vertices shared one dictionary at one address in memory. Need to comment through the Mapping file and get back to work on testing.
* **3/12/2020:** Testing system now has a working Vertex-to-Territory system, can soon start working on the test game dictionary and testing nation abilities and functions.
* **3/13/2020:** Test Game can build projects, finish them and apply their effects. It can also deduct maintenance and resource costs as well as gain resources and income over time.
* **3/14/2020:** Test Game can now buy armies, but money is not going down so we need to see about that.
* **3/15/2020:** Money growth has been fixed!
* **3/16/2020:** Rewriting most of the project to use less savegame data and to fix major bugs.
* **3/23/2020:** Re-familiarized myself with the bot, set some short-term goals.
* **3/27/2020:** Finished project buying and placement in target territory, set up some basic GameObject features like status processes.
* **3/30/2020:** Finished basic onNewTurn for game objects and territories, need to construct entire NewTurn and NewYear systems.
* **4/11/2020:** Added more to skeleton onNewTurn function, buildings now finish building when their completion date has passed.
* **4/16/2020:** Need to define onNewYear results. During a new year, onNewTurn processes will be done. In addition, demographics will change size and unrest levels, nation will get resource income, loans will demand payment. This is all represented by an onNewYear dictionary, which doesn’t have project effects for that turn (that’s handled by onNewTurn), instead it returns a dictionary of all resource incomes, maintenances, etc.
* **4/16/2020 cont.:** onNewYear is going nicely, but just remembered why newYearEffects is a good field for a gameObject: It saves revenues, incomes etc. and therefore saves calculation time. Change newYearEffects when a project is added, changes status or is removed, or if a Demographic changes. Basically, change newYearEffects whenever an event occurs in the territory.
* **4/17/2020:** Did more onNewYear work. Territories now send resource revenue and maintenance costs up to their owner nation, and those are processed by the nation. Also, when adding a nation to a savegame, they automatically get modifiers for all of the defined resources in the savegame’s dictionary. Also, ecided against newYearEffects as an attribute. Game is not constantly running, so a short delay is fine for new year events.
* **4/20/2020:** Started on the basics of the UnitGroup system: If possible, a Unit Group is made with a composition consisting of exactly one division. Also fixed a bug that made it so that bureaucratic space didn’t go down after a project was completed. Need to consider modifying applyEffects such that bureaucratic efficiency modifiers are added to the actual values rather than just kept in modifiers.
* **4/22/2020:** Added some more UnitGroup stuff, mainly a decrease in the local population. Each demographic loses a number of population based on its local recruitable population. The recruitable population calculation is something I’m going to work on even further, taking into account things like sex ratio.
* **4/24/2020:** Started on map-to-imagefile method (toImage) and its working well! For future reference: Maps should only show territories, ownership, edges and possibly names/ID and terrain. All other information should be in text files.
* **4/30/2020:** Finished the pathfinding algorithm! Map can use A\* algorithm to calculate the shortest path between vertices, as well as print an image of the path if it wishes to. I do need to be able to make the path image overlay on top of the original map, but other than that it works perfectly!
* **5/7/2020:** Decided on the inclusion of time-based (“Temporary”) modifiers. Also decided that for now, demographics will only gain unrest if they are at a deficit of resources, a building in their territory is foreclosed or the country is at war, and will lose unrest if a new building is built, more resources than they need are coming in or in storage.
* **5/11/2020:** When an object is disabled from lack of resources, its name (and territory, if applicable) will be stored in a temporary modifier key with the format below. The object will then get the “Disabled” status and not send in its maintenance costs as long as it is disabled. If the nation automatically gets enough resources to afford at least one of the disabled objects again (including gaining enough resources for a re-enabling before a new year begins), then it will enable the costliest object that is less expensive than the amount regained..

Nation.modifiers[“National”][“Temporary”][f”{resource}-lacking objects”] = {f”{territory}:{projectName}”: resource cost, f”ARMY:{projectName}”: resource cost}

* **5/27/2020:**
  + Will add a new Settings parameter to the Nation object including user-toggled settings, such as “Armies prioritized for disabling”.
  + Choosing which objects to disable works like this: A list of projects and unit groups is compiled that all require a resource that is in deficit. More expensive projects will be disabled before less expensive projects, projects that bring in little or no revenue will be disabled before projects that bring in a large proportion of revenue, smaller unit groups will be disabled before larger unit groups, and more resource-intensive unit groups will be disabled before less resource-intensive unit groups. If a unit group requires a resource that is not available, the division of that group requiring that resource will be split off and disabled, or “abandoned”. Whether Projects or Armies are disabled first is set by the player.
* **5/28/2020:**
  + Began real work on disabling algorithm. So far, added loop to delete objects with break conditions, have empty loop that will add all disable-able objects to a dictionary.
* **5/29/2020:**
  + Did some work on loop that adds disable-able objects to dictionaries for processing
  + Realized that Unit Group maintenance is not counted
* **6/1/2020:**
  + Fixed some bugs, including fixing Unit Groups such that they now become active after they are completed with training.
* **6/5/2020:**
  + Did a little bit more work on disabling algorithm. It now successfully adds projects and armies to the disable-able BinDescTree, which is automatically sorted due to how it works as a data structure. All I need to do now is to disable the projects one by one.
  + Created a new data structure in Util.py, BinDescTree. Pretty much combines a Binary Tree with a Dictionary: every node not only has a numerical value, but a “descriptor”, which is kind of like a key.
  + Technicals:
    - Projects will be sorted for disabling priority first by how expensive they are, then by how little revenue they bring in. This is demonstrated by the equation projectWeight = (cost - (cost/avgRevenuePercentage))
    - Unit Groups will be sorted for disabling priority first by how small they are, and then by how expensive they are in that resource. This is demonstrated by the equation unitGroupWeight = (size – (resourceCost/size))
* **6/6/2020:**
  + Finally finished the resource deficit consequences functions (outside of monetary loans)!
  + Fixed Nation.trainForce() so that it doesn’t exclusively create unitGroups.
  + Need to make sure players can’t cheese re-enabling projects. Basically, once a project is re-enabled, it must wait a turn before it can start applying its effects.
  + Need to make sure that modifiers are lost if a project is disabled.
  + Need to increase maintenance with unit size!
  + Need to be able to split off undersupplied units from a larger unitGroup. Also, think about splitting off ships based on health.
* **6/17/2020:**
  + This time, I have a good reason for my break: I was programming a battle program for the “Conquest and Imperialism II (1444)” server!
  + Made a new kind of process for GameObjects called Countdowns: instead of “processName:completionMonth/completionYear” it looks like “countdownName:x Turns” where ‘x’ is an integer number.
  + It seems I’m smarter and more prepared than I thought: I have the capacity to expand Nation.disableObject() such that I can subtract project effects. Need to actually do that, though
* **6/21/2020:**
  + Got back to work after a short bug fix on the battle program
  + Finished Territory.retrieveEffects(), need to test with territories that have projects.
* **6/23/2020:**
  + Fixed some minor bugs with Territory.retrieveEffects() and Nation.applyEffects(), tested Nation.annexTerritory() with territory that has project, works just fine.
* **6/24/2020:**
  + Finished theoretical code for disabling individual units within an army, need to test, probably with a brute-forced blueprint.
  + Units now scale maintenance costs with size. Accordingly, I changed unit costs to reflect costs for each 1 atomic part of a unit.
  + Need to do effect removing for projects, etc.
* **6/25/2020:**
  + Successfully tested unit disabling! Also made some revamps to GetNewForceNum, namely accessing it through the function GetNewForceName that automatically returns a Unit Group name instead of just the number.
  + May want to revamp scaling maintenance with unit sizes, as well as combining getNewForceName and getNewForceNum
* **6/26/2020:**
  + Removing effects works! Implemented this from both projects and territories.
  + Started defining Loans and the different subclasses (so far, indefinite and timed)
  + Made it so ceding territories removes disabled object names from the national list, and annexing territories adds their disabled objects to that same national list.
  + Changed “Territory.readyForNewProcess() to Territory.isActive()” for clarity and used it to make sure disabled project effects are not added to the Territory.retrieveEffects() return.
  + May want to make it so that objects check for any processes at all in onNewTurn or onNewYear.
* **6/27/2020:**
  + Started on the real work for loans. Nations now pay off loans over time automatically for the minimum annual payment required, or adds strikes to each loan if it cant. I also made an empty Nation.forclosures() method for deleting projects.
* **6/30/2020:**
  + Have been doing very minor work with loans, I need to actually define what foreclosure is and what happens during it.
* **7/2/2020:**
  + Made foreclosures result in temporary unrest modifiers and the removal of foreclosed object effects
  + Made loans automatically repayed or foreclosed based on the strikes and standing of the loan
  + Need to do national credit and loan defaulting
* **7/3/2020:**
  + Fixed a lot of bugs with loans.
* **7/4/2020:**
  + Made interest equation! Nations can now calculate their interest rate with Loan.calculateInterest().
    - interest(x) = (0.000003 \* (credit^2)) - (0.0012 \* credit) + 0.01
    - interest(-100) = 0.25, interest(0) = 0.10, interest(100) = 0.01
* **7/9/2020:**
  + Loans now can default if there is nothing to foreclose for them!
  + Loans now increase and decrease the national credit score whenever they’re paid out.
  + Added a function to get a nation’s revenue at the present date to calculate the rough size of the economy.
* **7/14/2020:**
  + Finished making moving statuses as well as allowing unitGroups to move. Just need to take into account the kinds of territories they can move into.
* **7/17/2020:**
  + Started work on sea provinces and fleets.
* **7/18/2020:**
  + Started work on constructing fleets. Renamed “SubFleet” class to “Ship” so I can easily use the Nation.trainForce() function.
  + Need to be able to build ships in coastal Land provinces.
* **7/19/2020:**
  + Building fleets is completed!
  + Fixed some bugs with unit building, specifically making bureaucratic usage lessen when training is done.
* **7/20/2020:**
  + Made a new test map from scratch using MS Paint and a new function that translates pixels into coordinates. Will be able to use it to test fleets and the like.
* **7/21/2020:**
  + Finished Fleet.embark() and Fleet.transportCapacity()!
  + Added “Weight” to units in Game Dictionary. Basically, each unit takes up the space of its size times its weight. 100 Cannons with a weight of 10 take up the same space as 1000 Infantry with a weight of 1.
* **7/23/2020:**
  + Finished Fleet.disembark()! Tested both embarking and disembarking as well!
  + Changed Fleet.onNewYear() to include embarked unit group onNewYear()s!
* **7/24/2020:**
  + Made “RelationalMatrix” and “RelationSet” objects to hold diplomatic relations between nations.
* **7/25-26/2020:**
  + Created Expedition UnitGroup, UnitList Unit subclass, and Specialist and Colonist Units.
  + Can recruit colonists from a territory now.
  + Unrest affects the recruitable population of a territory.
* **7/28/2020:**
  + Can now colonize provinces! Expeditions can move between provinces and settle on uncolonized land.
* **7/29/2020:**
  + Made it so some more modifiers are counted in for nations and territories and their constituent objects.
* **7/31/2020:**
  + Added “HabitableTerritory” and “ColonialTerritory” classes in order to use an object-oriented approach to differentiate between uninhabited, non-colonial inhabited, and colonial provinces. Colonial Territories now actually cost money based on the population rather than getting taxes.
* **8/1/2020:**
  + Fleshed out “HabitableTerritory” and moved all demographic functionality from Territory.
* **8/3/2020:**
  + Made Army and Fleet cost modifiers count
  + Defined test colonial buildings for colonial progress testing
* **8/5/2020:**
  + Made the rudimentary Colonial Buildings and Colonial Progress system! I don’t expect much has to be done more on the former front, considering the building effects system that already exists.
* **8/9/2020:**
  + Started work on trade and diplomacy! Made diplomaticRelations for saveGames, mailbox for Nations, and a very basic trading procedure. Need to test it!
* **8/10/2020:**
  + Finished the trading system, including trading diplomatic relations!
* **8/11/2020:**
  + Started organizing the code into packages, learning how python packages work.
* **8/12/2020:**
  + Finished organizing the code so far into a package!
* **8/21/2020:**
  + Over the past 9 days, I’ve made NationController (a file that defines lots of functions for the bot to interact with the game code) and have made maps that are colored by nation and labeled by territory name.
* **8/30/2020:**
  + Over the past 9 days, I made the actual bot work in some limited capacity, including error handling and logs, savegame and player caches and some embeds.
* **9/4/2020:**
  + The bot now can visualize national info, armies, and go back to a previous menu!
* **9/6/2020:**
  + Did a LOT of error wrangling and code restructuring, particularly with making NationController.nationalInfo return a dict of objects. This is because if I try to saveObject the return output then it turns all listed objects in the nation objects into dicts as well.
  + Made the bot display individual armies!
* **9/7/2020:**
  + Made the bot show a list of territories in the nation, as well as an army’s information.
  + Fixed an error where exceptions were improperly handled when printing player stacks to a file, since embeds couldn’t be JSON serialized. I made it so that embeds in lists of pages were not included in a printed stack, but that instead stacks only held the commands that were used with them.
* **1/3/2021:**
  + Returned to development after a few months of school!
  + The entire project has now been uploaded to, and is continually updated on, github.
  + The bot can now display all buildable projects, all projects in a territory, and all information for one project in one territory.
  + The bot can now display national and territorial demographic makeups, as well as all information for a specific local demographic.
  + Display is almost done! Just need to make maps actually readable without any province name overlaps.
* **1/12/2021:**
  + Display has been done for a few days, worked primarily on Unity learning for some time.
  + Made a function called “validateArgs”.
    - Discord commands will interpret the arguments for the command “n.build Big Castle New Delhi” not as “[“Big Castle”, “New Delhi”] but as [“Big”, “Castle”, “New”, “Delhi”].
    - To fix this, I run all args through validateArgs, along with a dictionary, where the keys are the names of variables required to perform a game function and the values are enums specifying the variable type or iterable objects that the keys need to be present in to be validated. I’m quite proud of it!
  + Finished the first gameplay commands, trainArmy and trainFleet.
  + Fixed a bug where in SaveGame.saveMapImage() it would iterate over all the nations in the saveGame, then use the name of the last nation iterated over in the subsequent command for only showing the territories owned by a specified nation.