Modern Tactics

Real Time Strategy (RTS) Game Kit

Flexible Real-time Strategy Game Kit

Unity 5 Ready!

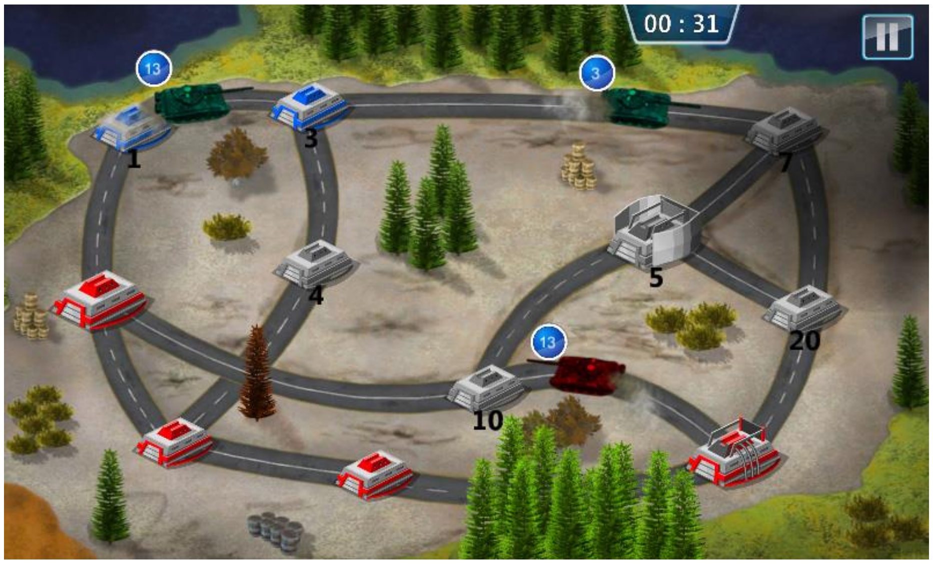
Supports Android, iOS, Windows Phone, WebPlayer, Windows and Mac

Dear Customer,

Thank you so much for purchasing this game kit. Here you can find detailed information on how to use this product with maximum proficiency. All scripts and code assets are fully commented, but if you ever needed a hand on a segment of the codes or anything else, feel free to contact us at http://www.finalbossgame.com . We'll try our best to support you with your questions as soon as possible and till you feel most satisfied.

Overview

Modern Tactics is a massive, carefully designed and complete real-time strategy game kit at your finger tips. The kit features a detailed implementation of a modern RTS game with all required game mechanics like base management, troops commanding, war tactics (including both defensive and offensive tactics), base capturing, ground and aerial combats and an unbelievably genius AI system that controls the enemy side of the game, and can engage both player and other enemies!



This project works flawlessly with both touch and mouse inputs, and thus, can be tested on **Android**, **iOS**, **WebPlayer** and **Stand-Alone** platforms simultaneously. The kit also works on both Unity3d Free and Pro, and again to your benefit, needs no 3rd party plug-in or add-on to work.

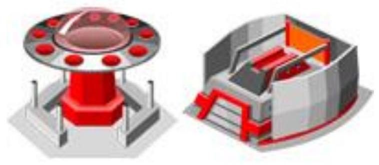
It just works ***right out of the box***. You do not have to worry about anything.

!Important:

- When you load the kit for the first time, always add all scenes to the "Scenes in build" list via ***file->BuildSettings***, to make sure you experience a smooth transition and game flow.
- The game requires these Tags to work:
  - Base
  - Deactivator
  - Troop
  - ground
  - explosion
  - musicPlayer
  - GUIBtnHolder
  - GUIClock

  Make sure to maintain them in your personalized project or replace them with proper equivalent.

- The kit uses 3d objects with their Y scale set to very small float (0.001) to simulate the 2D feel of the game. This way you have the freedom to easily extend the kit to 3d by replacing all default 3d cubes with your own 3d objects, and restore the Y scale to your custom settings. However you can maintain the current flat 3d structure and use your 2d textures and sprites to decorate the game environment.



These are 2d textures applied to flat 3d cubes!

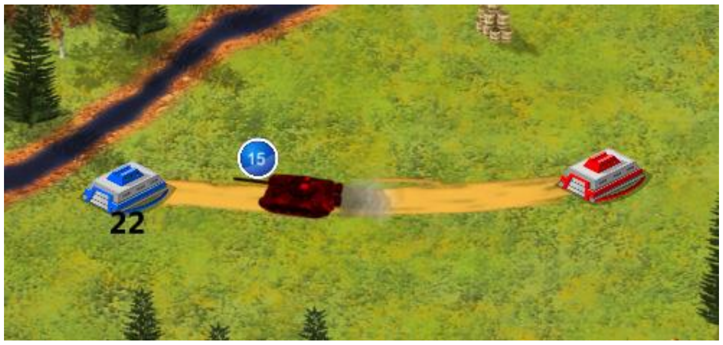
- As the kit uses 3d objects, you have to carefully set the Y position of each element in the scene. As a rule of thumb remember to always position different game objects with this rules:
  - Y position of all **Bases** should be set to 0.2
  - Y position of all **Roads** should be set to 0.1
  - Y position of all **Paths** (Main Parent) should be set to 0.21
  - Y position of **winPlane** should be set to 5.0
  - Y position of **_PauseSystem** should be set to 3.0
  - Y position of **nowLoading** should be set to -3.0

- You need to have all these 7 controllers in your custom levels. These are mandatory and must be present.
  - ___EnemyMasterMind
  - ___OutlanderMasterMind
  - ___GlobalTouchManager
  - ___SceneGraph
  - NowLoading
  - _PauseSystem
  - winPlane

Game Play

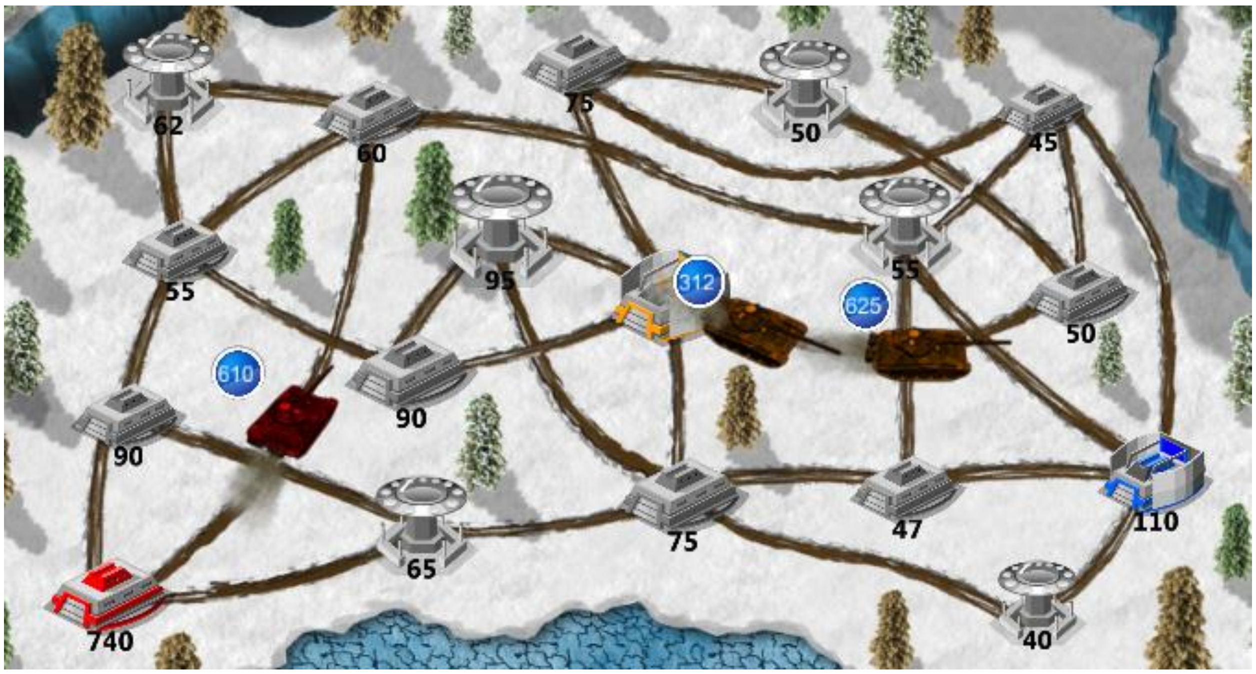
This kit uses some unique and advanced strategy tactics and this makes it a brand new experience for your players. In this game, you play as a commander, and have one or a few bases at your side. On the other side, you have at least one enemy (AI Controller) that has its own bases. You have to defeat your opponent by capturing all its bases and destroying all its troops.

You can command your bases to send troops to other bases. You are able to select half or full amount of troops you have inside your selected base, and then you can issue the command by selecting the destination base, and when done, your troops begin to move to their destination. Upon arrival, they will engage with troops inside the destination base. If they are stronger than the defending base, they will capture the base for you. Otherwise, they will kill as many units inside the target base that their attack power and target base's defense modifier (fortification factor) allows.



Enemy AI is attacking a player base with 15 tanks. Player base has 22 defending units inside, so the attack ends with attacking tanks getting killed completely, while player base remains with 7 units inside.

The game features different types of game-play like campaign and endless, with complex level structures, multiple AIs, Both Ground and Aerial bases in one scene, fast growing bases (with more than 500 troops) and different environments. These options help you to build a feature rich game play that never disappoints your players.

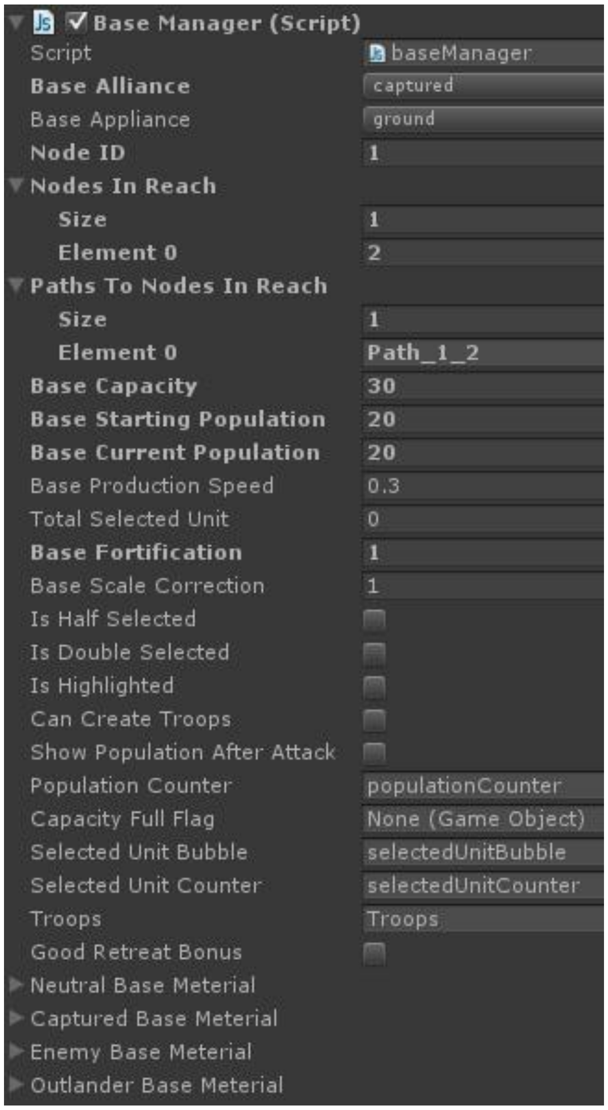


Game-Play Mechanics

There are a total of 4 sides in this game, including the **Player**, **Enemy** (AI-1), **Outlander** (AI-2) and **Neutrals**. Neutral bases have no owner and are available in game for other sides to capture. Other bases with owners produce troops as the time passes in the game.

Each side has two types of bases, including **Ground** and **Aerial** types. Ground bases produce tanks and aerial bases produce planes. Tanks need roads (paths) to move between different bases in the scene, while planes need no road to move and can freely travel (Attack/Defense) to any base.

Bases come in different shapes, sizes and attributes.



Base Manager (Script)

Script	baseManager
Base Alliance	captured
Base Appliance	ground
Node ID	1
Nodes In Reach	
Size	1
Element 0	2
Paths To Nodes In Reach	
Size	1
Element 0	Path_1_2
Base Capacity	30
Base Starting Population	20
Base Current Population	20
Base Production Speed	0.3
Total Selected Unit	0
Base Fortification	1
Base Scale Correction	1
Is Half Selected	☐
Is Double Selected	☐
Is Highlighted	☐
Can Create Troops	☐
Show Population After Attack	☐
Population Counter	populationCounter
Capacity Full Flag	None (Game Object)
Selected Unit Bubble	selectedUnitBubble
Selected Unit Counter	selectedUnitCounter
Troops	Troops
Good Retreat Bonus	☐
Neutral Base Meterial	
Captured Base Meterial	
Enemy Base Meterial	
Outlander Base Meterial	

Each base has a side (Base Alliance).
Each base has a type (Base Appliance)
Each base uses a unique node ID to be indexed inside the scene.

Ground bases can only reach to their neighbor bases with roads. “NodesInReach” caches the nodeID of all available neighbors for this base. Aerial bases don’t need this array.

For each available neighbor, we have to carefully design and set a path object, so ground units can grab and follow it to reach their targets. It is possible to use no path for the bases, but in that case, ground units always follow a direct line between starting and target base. This is especially useful when you want to design games with macro-management strategy, like “Age of Conquest”.

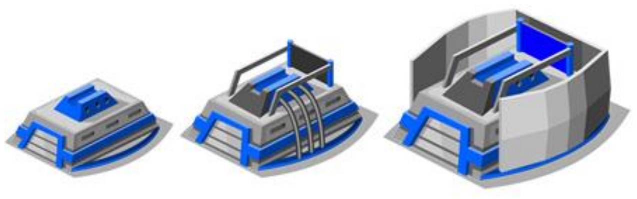
Base capacity indicates the maximum number of troop a base can create before going to sleep. If you order some troops to leave the base, the controller starts to create new units again.

Each base has a starting population at the beginning of the game. This value can exceed base capacity.

You can also set how fast a base can produce its units by changing the production speed.

Fortification is a defensive bonus for all bases and can be set to 1, 2 or 3. Their defensive power will be multiplied with the fortification factor. For example, to capture a base with 10 units and a fortification level of 2, you need 21 attacking troops. 20 to kill (10*2) defending units and 1 to capture the target base.

Refer to “BaseManager” class for detailed instruction on how to control base attributes.



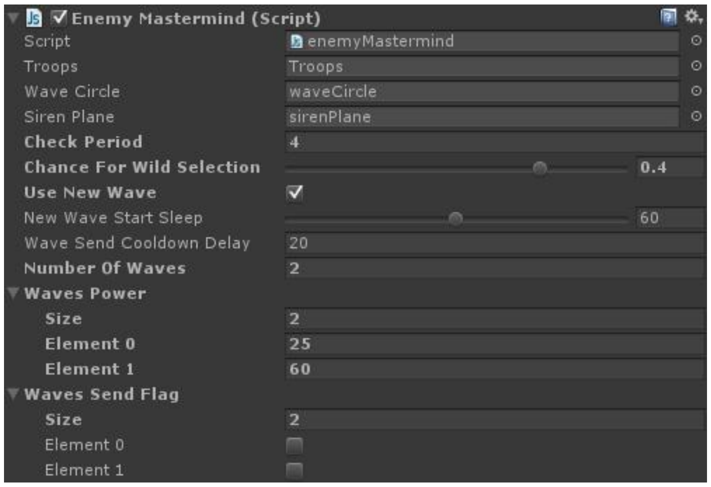
Different shapes of player’s ground base with different levels of defense (Fortification Factor)

Introduction to Scripts, Classes and GameObjects

All scripts in this kit are fully commented and have full description about what they do. But here we take a closer look at some more important classes used in the kit.

enemyMastermind

This game object alongside its class, is the main controller for AI in the game. Enemy mastermind monitors game status and uses its decision making process to perform each moves. You can modify general aspects of this controller like the decision cycle delay (how fast enemy can monitor the game and perform actions), the wildness factor which is used when we intentionally want enemy AI to perform irrational moves to fool the player. You can also define backup waves for enemy in every level you want. You can simply check "UseNewWave" option and then set the parameters to handle wave creation.



Enemy Mastermind (Script)

Script	enemyMastermind
Troops	Troops
Wave Circle	waveCircle
Siren Plane	sirenPlane
Check Period	4
Chance For Wild Selection	0.4
Use New Wave	✓
New Wave Start Sleep	60
Wave Send Cooldown Delay	20
Number Of Waves	2
Waves Power	
Size	2
Element 0	25
Element 1	60
Waves Send Flag	
Size	2
Element 0	☐
Element 1	☐

If we want to use more than 1 backup wave, we can set a cool down timer to make a delay between comings of each wave.

Then we should indicate the number of waves, modify WavesPower array with the number of waves we set and indicate each new waves power.

At the end, we just need to set WavesSendFlag array size to total number of waves.

***Outlander** mastermind is a direct clone of enemy mastermind class and features some small changes that you can check for yourself.*

However make sure to set correct state for "OutlanderIsInTheScene" Boolean variable in your custom levels. It tells other controllers if there is an Outlander in the scene. So if you want to have a second enemy (Outlander as the second AI), check this box, and if you want just one enemy AI, make sure to leave it unchecked.

globalTouchManager

This class is responsible to manage all click/touch events inside a live game. It remembers player's interaction with different bases and handles selecting, de-selecting, and command issues inside the game. This class also allows baseController class to create and send troops.

sceneGraph

SceneGraph acts as a data provider for other classes. It provides static functions that check game conditions, find nodes, paths, and checks if two nodes have direct access.

GameManager

GameManager simply monitors game status to detect if the game is finished (Win or Lose) and also is used to save game play time.

winPlane
WinPlane controls the statistics screen you see after you win or lose a battle. It handles user input on its buttons, and also provides useful data gathered inside the game and show them to user as statistics.

RankManager
This kit uses an internal leveling (experience) system and every time player wins or loses, gives him some experience points based on his performance in the battlefield. RankManager uses this experience parameter to grant user new ranks, as the player advances in the game. You are free to use this rank system to provide new additional levels or items to the player. In addition to XP, we have provided you with a simple money prize you can grant to your players, so they can buy additional items in your game. (We are already working on the next update of the kit to bring new features including in-game shop system, power ups, special items, and achievement system).

PathManager
PathManager connects two different ground bases to each other. Each path has a NodeID for starting point and a NodeID for the destination. You have to carefully set the name of the path object (in order to set it in bases inspector) and their "From" & "To" parameters, to connect the two bases. Each path has 6 transform as its children that you have to carefully reposition them to make a way between two bases. All ground units follow these waypoints in their journey to the destination base.

Important: Please bear in mind that Paths should always have a Y position of 0.21, while all their child transform objects should have a local Y position of 0. The X and Z parameters for rotation of all path objects should also be set on 0 (you can rotate them around Y axis with no problem).

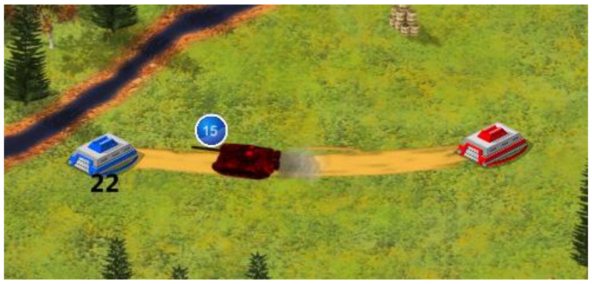
EpisodeSelectionManager & ButtonStateManager
This class handles player's advancement in the game. You can design and add as many levels as you wish, and you can make them available with or without order. If you want to have some sort of order for the levels (in case you want to use a story for your game) and open new levels upon completing the previous ones, all you have to do is to carefully name your levels, so this controller be able to save your players progress in the game. You can indicate if you want all levels opened with no restriction, or you want to open them by beating the previous one, by setting correct value for "**useLockSystem**" in ***ButtonStateManager*** class.

TroopMover
This is the heart of the game. This class handles all thing relates to troops and their behaviors and their interactions with the game objects in the scene. It handle troops movement, direction, path following, material, color and texture settings, engagement with other bases, and different battle sequences. It's important to know that troop prefabs doesn't exist in scene on their own. They are always instantiated by other controllers, because they require some of their parameters to be set at the moment of creation. They need destination, starting point, a path to follow (for ground units), movement speed, power (number of units inside this troop), side, type and correct material and texture.

Troops can be Ground (Tanks) or Aerial (Planes). Ground units need path to follow to reach to their destination base (Although you have the freedom to do not provide a path, which in case they follow a direct line between starting base and target base), while Aerial units doesn't need path to move and can get to every base in the scene with no limit.

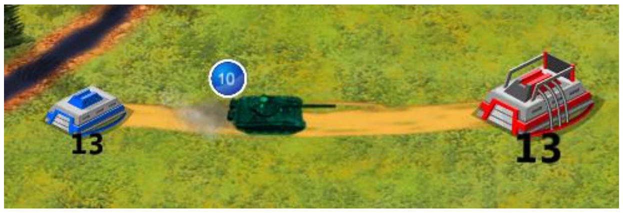
One important factor used in the game is that although you can attack a ground base with planes and also can attack an aerial base with tanks, but you will suffer a penalty of 2x casualty in your attacks, because of the different type of attacker force and defender base. This is a huge strategy implementation, and adds more depth into game play experience, because player has to think twice if it's wise to attack a base with different type or not.

The other important factor is the target base's fortification factor. Bases with a fortification factor of 2 or 3(max), act two or three times stronger when they are getting attacked. Look at these pictures for a better explanation:



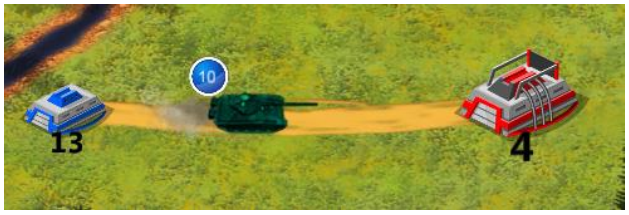
Attacking a ground base (with normal fortification level 1) with ground troops.

Result: 22 – 15 = 7 (attacker dies and defender base remains with 7 units inside)



Attacking a ground base (with fortification level 2) with ground troops.

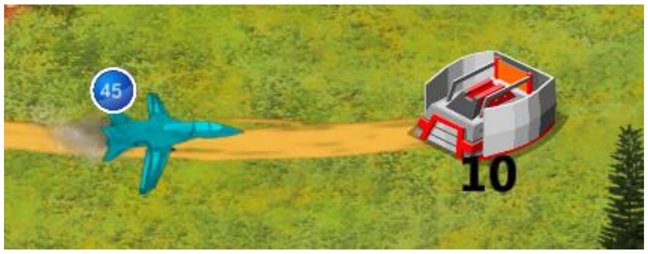
Result: ((13 * 2) – 10) / 2 = 8 (attacker dies and defender base remains with 8 units inside)



Attacking a ground base (with fortification level 2) with ground troops.

Result: (4 * 2) – 10 = -2 < 0

defender base is captured by attacker and its new population will be Abs(-2) = 2



Attacking a ground base (with fortification level 3) with Aerial troops.

Result: (((10 * 3) * 2) – 45) / (3*2) = 15 / 6 = Ceil(2.5) = 3

attacker dies and defender base remains with 3 units inside

please take note that how an attacker with 45 attack power cannot capture a base with just 10 defending units! (This is the power of strategy and good use of ***type mismatch*** and ***fortification factor***)

How to add new levels to the kit?

There is two way of creating new levels. The easy way is to clone one of the available blank templates, and build your own level on top of it. The other way is to start from scratch and create everything on your own. In both cases, you must have all the required controllers in the scene (we have discussed it before in page 3). Here we use a step by step tutorial on how to add a brand new level to the kit, create some bases, paths and then use a setting to make it all work together.

To add a new level to the kit:

1. Select and clone the “Blank-Level-Player-vs-AI-Path-01” scene by pressing Ctrl + D.

2. Rename this new level to something appropriate. Make sure to have an integer as a part of the level name. This helps other controllers to track and save player progress inside the game. Something like “mission-01” will be a good example.

3. There are already two bases (Red and Blue) + two paths (Red and Green) + 1 road object inside the blank template. Now you have to design your new level. We strongly recommend you to do this on paper, by thinking about all possibilities, bases, positions, fortifications, paths, waypoints, and bases starting populations. Look at this example:

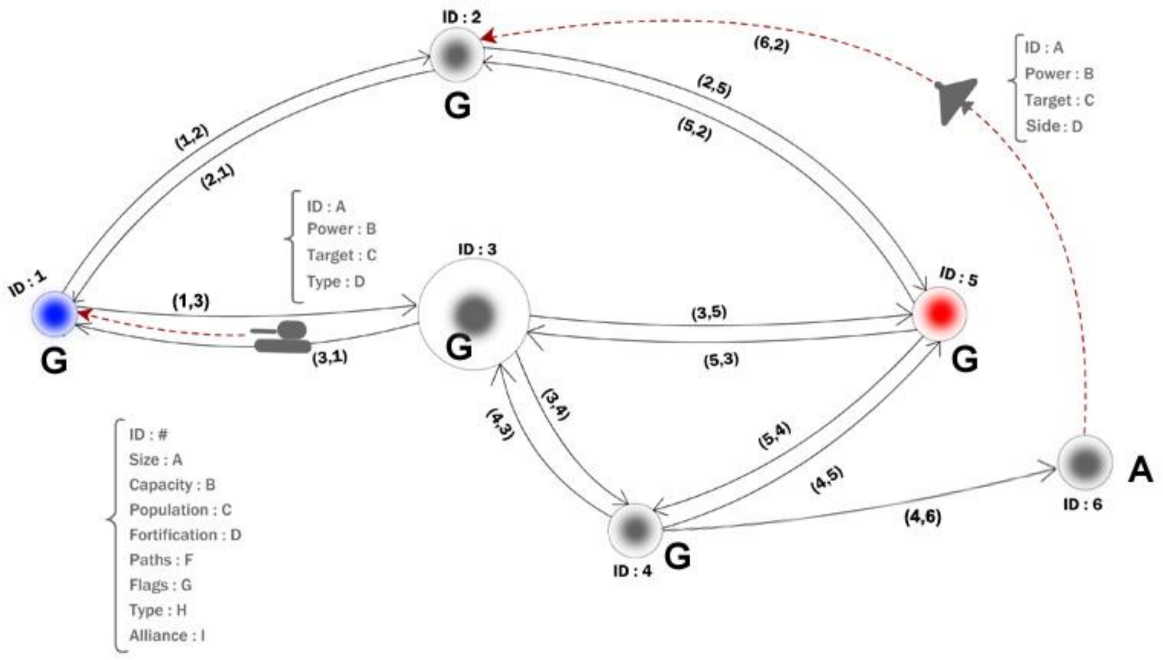


Figure 1 – Now you know what your final level will look like!

4. After designing your level on paper and completing the sketch, you know what you will have in your level. In our case, we have a total of 6 bases (5 grounds and 1 aerial). We have 1 base for player (Blue dot) and 1 base for enemy AI-1 (Red dot) and we also have 4 neutral bases. We have no second enemy (outlander). We have 13 paths and we also need 7 roads to cover the paths.

5. We begin by setting up our bases. First clone one of the available bases (or you can drag base prefab into scene) and repeat this till we have 6 bases in the scene.

6. Now carefully reposition all these bases to match our initial design. Don't worry about bases colors, textures, sides and NodeIDs now. We will fix them later. Always remember that all bases should have their Y position set to 0.2.

7. Now set each bases nodeID based on our initial design. We should have 6 bases with nodeIDs from 1 to 6. To do this, select each base and enter its ID in nodeID field. This is unique for each base.

8. Then select base with nodeID = 1 and set its base alliance (side) to captures. This will be player's starting base. Then head over to mesh renderer component of this base and select "capturedBase_01" as its material. Now it looks like a normal ground base which belongs to the player.

9. Simultaneously select bases with nodeIDs 2, 3, 4 and set their base alliance (side) to neutral. Then head over to mesh renderer component of these bases and select "neutralBase_01" as their material. Now they look like a normal neutral ground base.

10. Select the base with nodeID = 5 and set its base alliance (side) to enemy. This will be enemy's starting base. Then head over to mesh renderer component of this base and select "enemyBase_01" as its material. Now it looks like a normal ground base which belongs to the enemy.

11. Select the base with nodeID = 6 and set its base alliance (side) to neutral. This will be the only aerial base in the scene which only has road access from nodeID 4. Head over to mesh renderer component of this base and select "Aerial_neutralBase_01" as its material. Now it looks like a normal aerial base which is neutral.

12. Now we should setup paths between bases. Select player base with nodeID 1. This base has access to two neighbor bases with nodeIDs 2 and 3 (with paths (1,2) and (1,3)). Se we must indicate 2 paths for this base. First set "NodesInReach" array size to 2, and then enters 2 and 3 in the element 0 and element 1 field of this array. It tells the base that it has access to two neighbors with nodeIDs 2 and 3. For now, leave the "PathsToNodesInReach" array as is, because we have not created the paths yet.

13. Follow step 12 for all other bases in the scene. Please remember that you do not need to set "NodesInReach" array for aerial bases. So for nodeID 6, leave "NodesInReach" and "PathsToNodesInReach" with size 0.

14. Now we want to build the paths. We know that we need a total of 13 paths in this level and we already have 2 paths from the blank level template. So select one of these paths and clone it (Ctrl + D) till we have exactly 13 paths in the game (or you can drag the Path prefab into the scene).

15. No you have to rename all the paths based on our initial design. We need paths with the following names:

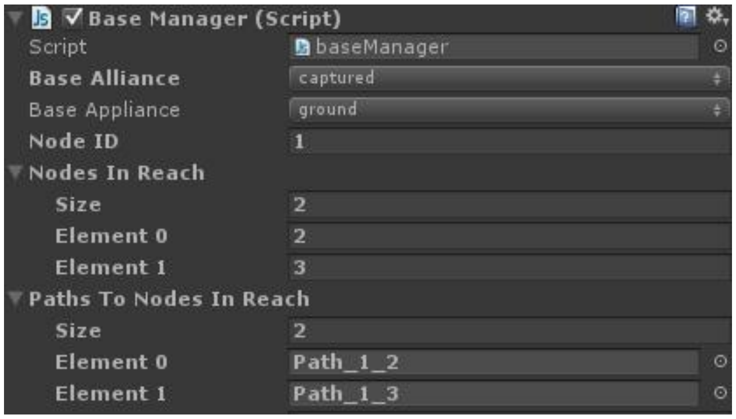
1. Path_1_2
2. Path_1_3
3. Path_2_1
4. Path_2_5
5. Path_3_1
6. Path_3_4
7. Path_3_5
8. Path_4_3
9. Path_4_5

10. Path_4_6
11. Path_5_2
12. Path_5_3
13. Path_5_4

16. Then select these paths individually and correct their "FromNodeID" and "ToNodeID" parameters. For example if we have a path named "**Path_X_Y**" then its "*FromNodeID*" is **X** and its "*ToNodeID*" is **Y**. repeat this for all other path objects. You can also check "ReverseMode" for similar paths (ie: path_x_y & path_y_x) so you can easily recognize them from their colors.

17. Now we have to carefully set the position of each path. Every path object has 6 child waypoint named from 1 to 6. You have to position the path so its first child object is on the "FromNodeID" base and its last child object is on the "ToNodeID" base. You are free to rotate the path parent object or reposition children within the path to make this right, but always remember that local Y position if the children should always be set on 0 and Y position of the parent path object must be set on 0.21. This procedure takes some time, so please do it patiently and check all paths and waypoints to have correct Y values.

18. After you are finished with the paths, we have to set them on bases. Select base with nodeID of 1, and set its "pathsToNodesInReach" array size to 2 to accept two path objects. Then based on the nodeIDs you have set for the "NodesInReach" array, select the correct path for element 0 and element 1 of the "pathsToNodesInReach" array. In our example, they should be set like this:


Base Manager (Script)

Script	baseManager
Base Alliance	captured
Base Appliance	ground
Node ID	1
Nodes In Reach	
Size	2
Element 0	2
Element 1	3
Paths To Nodes In Reach	
Size	2
Element 0	Path_1_2
Element 1	Path_1_3

Follow the same procedure for all other bases in the scene.

19. We are finished with the hard part, and now come the easy part. Select each base in the scene and enter your desired parameters for the following parameters"

- Base Maximum capacity
- Base Starting Population
- Base Current Population
- Base Production Speed

Please bear in mind that if you want to change the fortification factor of the selected base, you have to select a proper material for it too. Although the baseController automatically does this for you in real time, but it will certainly causes headache if you do not do this manually inside editor.

20. Now you have to drag appropriate road prefabs into the scene to make the way between bases visible to the player. The path objects are only visible inside editor so this is an important step in designing your levels. You have access to many different textures for normal, dirt and asphalt roads which you can use in the game.

*** A word of caution:** using too many discreet road objects in one scene can drastically hit game's performance by increasing the number of draw calls. So it's wise to design the final background texture (Containing all textures, roads, buildings, rocks, etc…) in an image editing software and use the final background in your game.

21. Now hit play and enjoy your new custom level. Never forget that balance is the key in any strategy game, so be fair to your players and keep up the balance in your levels.

Final Word

If you have any questions, feel free to ask us at http://www.finalbossgame.com and we will get back to you as soon as possible.

Our Other Cool Game Kits

Stealth Action Game Kit
Unity Assetstore
WebPlayer Demo
Youtube

Restaurant & Cooking Starter Kit
Unity Assetstore
WebPlayer Demo
Youtube

Endless Space Pilot Game Kit
Unity Assetstore
WebPlayer Demo
Youtube

Snakes & Ladders Framework
Unity Assetstore
WebPlayer Demo
Youtube

Finger Soccer Game Kit
Unity Assetstore
WebPlayer Demo
Youtube

Monster Blaster! Game Kit
Unity Assetstore
WebPlayer Demo
Youtube

Real Estate Tycoon Game Kit
Unity Assetstore
WebPlayer Demo
Youtube