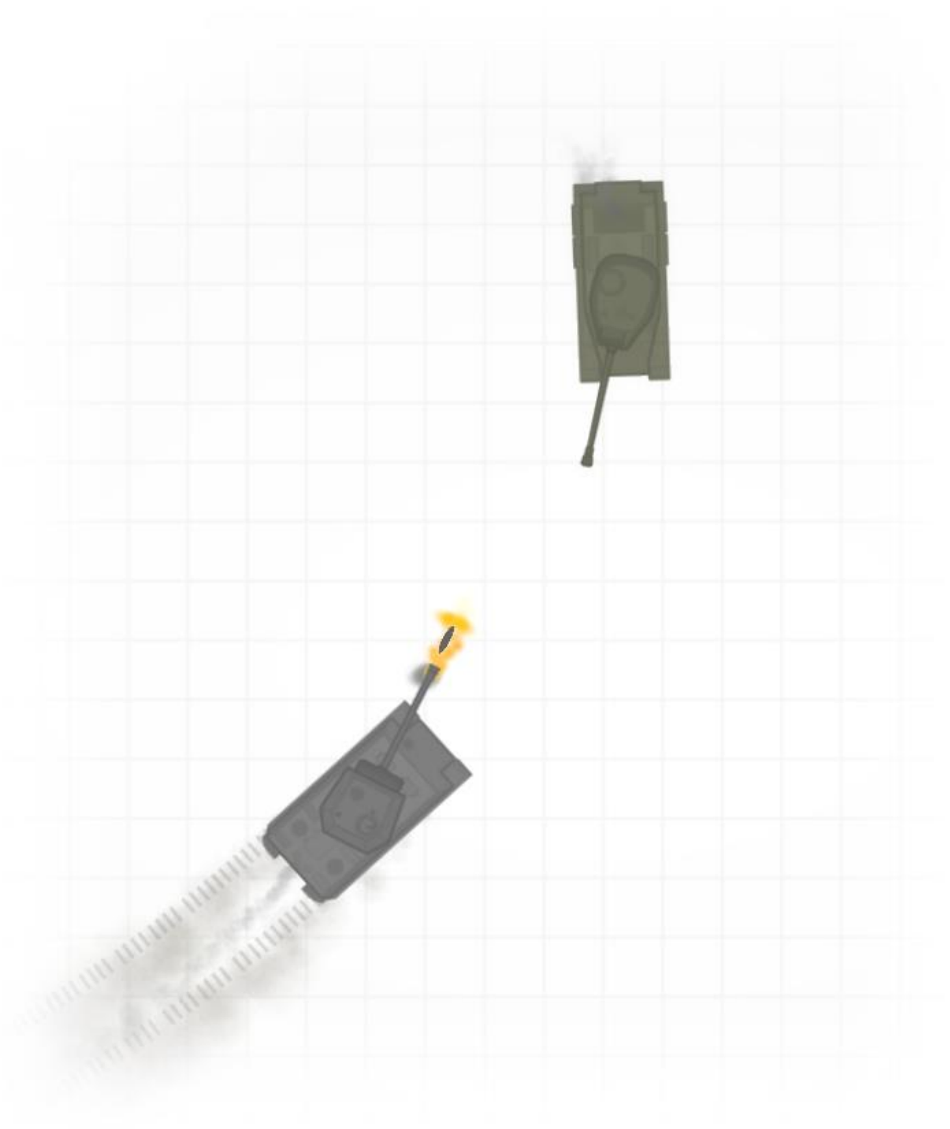
2D Tank Controller by ASL00

Documentation

For Asset Pack version: 1.0



Created: May 27th, 2018

Last Changed: -

For complete changelog see the last page

Thank you for buying the asset!

I hope you will enjoy using my asset pack!

This documentation explains all major features and mechanics of the asset pack and guides you through the setup process to make integration as easy as possible. This documentation also explains the process of making your own tanks.

In case you have any problems with this asset package, I'm going to do my best to help you to get it fixed as soon as possible. I'm currently studying, so my working hours are limited, but when I am working it is my top priority to fix any problems you may be having with this pack!

You can contact me via email at: contact.asl00@gmail.com

Also, to let you know, I live in Finland, so if you're located very far from Europe I may not be able to reply within the same day.

If you enjoy the asset pack, I would be very grateful if you posted a review on Asset Store as it would help me and others deciding whether to buy this pack greatly!

Also, if you have any feedback or suggestions feel free to send that to me too!

CONTENTS

1.1 | BASICS // Terminology

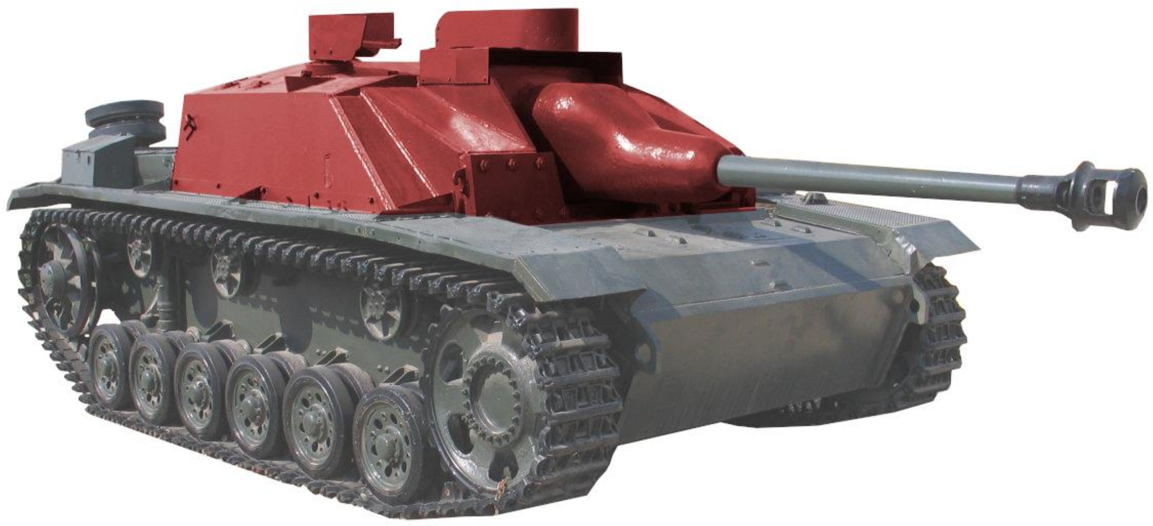
Many of the functions in this asset are divided into two layers. These layers are called the **hull layer** and **turret layer**. The armor model is divided into two parts, so the hull and turret have their own armor layout.

This also makes it possible to have partial cover so only the hull armor of the tank is behind cover, but the turret or the superstructure is vulnerable to incoming fire.

In this asset package and especially in the scripts **turret** means not only a 360° turret of a regular tank, but also the superstructure or the gun of a casemate tank destroyer.

For example, the superstructure armor of a tank destroyer should be on the turret layer, not on the hull layer, even though the superstructure is a fixed part of the hull and doesn't rotate. Also, in the script, turret traverse speed applies to both a conventional turret but also to the traverse speed of the gun of tank destroyer.

The red part is the superstructure of, in this case a StuG-III, tank destroyer:



1.2 | BASICS // Features

The 2D asset package has a ton of features, and the most important ones are listed down below.

TANK MECHANICS

- Input
    - Configurable input axes for tank driving, rotation, shooting and turret/gun rotation
    - Turret can be controlled with keyboard or mouse input
    - Hull rotation direction when reversing can be inverted
- Movement
    - Smooth and realistic acceleration rate
    - Fully configurable mobility stats (top speed, traverse speed, engine power, acceleration and braking force multipliers etc.)
    - Rotation Anti-Glitching System (rotation input will be disabled when forcing rotation when tank is stuck and unable to turn to prevent unexpected results)
    - Realistic tank collision physics and damage calculation
    - Tracking mechanics (immobilizing the enemy by destroying their tracks)
- Turret System
    - Both regular 360° turret tanks and tank destroyers with a fixed casemate and only moving gun are fully supported
- Sound Effects
    - Realistic sound effects include: engine idle & running sounds, track rattle sound, shooting sound, collision sounds, reload sound and tank explosion sound
- Particle Effects
    - Particle effects included: dust particles when driving, exhaust particles, shooting muzzle flash and smoke particles and tank explosion particles
- Track Marks
    - Tanks leave track marks on the ground for a small period and then they slowly fade away

SHELL MECHANICS

Shell mechanics are explained more in-depth on the next page. But here are the most notable features.

- Adjustable stats such as: caliber, damage, penetration and velocity
- Armor piercing (AP) and high explosive (HE) shell types
- Sound and particle effects for different events such as: penetration, bounce, ricochet, hitting the tracks or buildings

1.2 | BASICS // Features

BASE CAPTURE SYSTEM

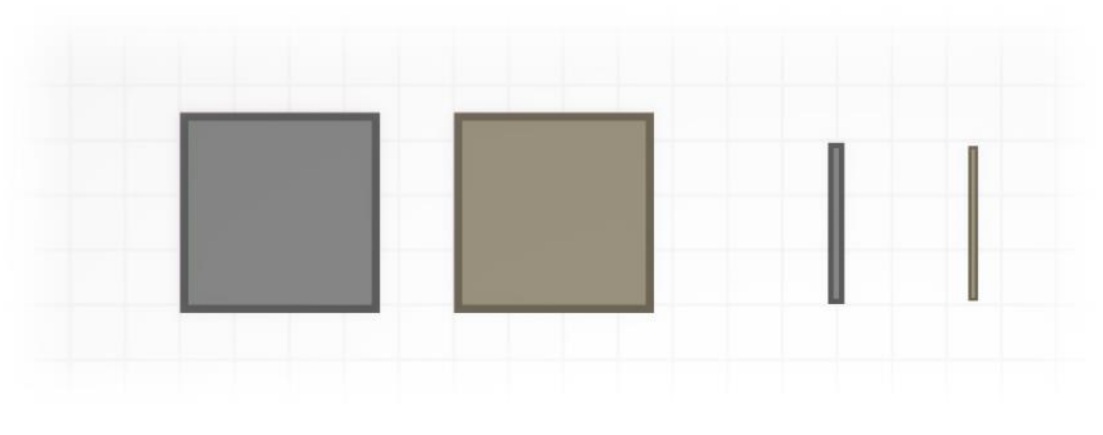
Although this asset package mainly concentrates on the tanks themselves, there is a foundation for a base capturing system. The capture zone prefab detects all tanks within its area, and if there is only one tank (alive) in the capture area, capture points variable in the Tank Controller script will be increased. This capture area also has some effects, so it changes its color and plays an alert sound when the base is being captured.

The script simply adds the capture points in the Tank Controller script, so it is up to you how the capture system is integrated into your game if at all.

STRUCTURES

There are four types of structures in this asset package. There are indestructible buildings and walls, and destructible buildings and walls. Indestructible structures are gray, and destructible ones are brown (wooden). The difference between buildings and walls, is that buildings are on both hull and turret layers, so they give full cover. Walls on the other hand only protect the hull of the tank, so the turret can still be shot.

Example buildings on the left and walls on the right.



Destructible walls and buildings can be destroyed by shooting or ramming, and their mechanics include:

- Configurable durability
- Damaged sprite (sprite that will be shown when a building or a wall has less than 25% “HP” remaining)
- Destruction sound and particle effects
- Destroyed sprite (sprite that will replace the original sprite after the building is destroyed)

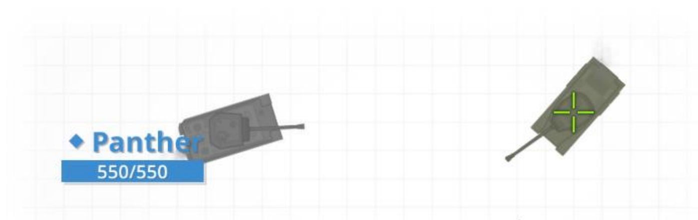
1.2 | BASICS // Features

UI FEATURES

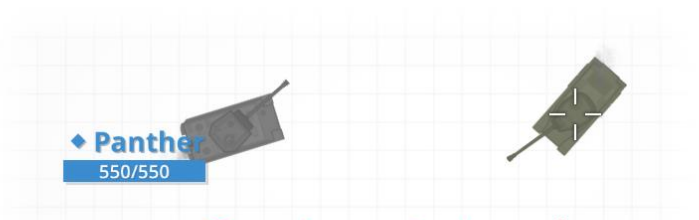
There are two main UI features in this asset package. One is the tank UI panel that displays the tank's name, type, HP, damage taken, reload time and the time remaining to repair the tracks, and the other is the crosshair.

The color of the UI can be changed, and its position can be clamped so that it doesn't go outside the map. And if you don't need the UI it can be disabled completely.

The crosshair is used when the turret rotation input is set to mouse aim. The crosshair changes its size and color depending on whether the gun is pointing to the direction of the mouse as shown in the images below.



Gun is aimed



Gun is not aimed

1.3 | BASICS // Shell Mechanics

One of the most prominent feature of the asset package is the advanced shell penetration and damage system. There are two shell types: Armor piercing (AP) and High explosive (HE) shells. Most of the mechanics are different between these shell types, but some mechanics are the same.

Next up is a detailed step by step explanation of the shell mechanics.

1 – Shell Creation

First the shell is created by the Tank Controller script. The Tank Controller knows whether the shell should go to the hull or turret layer (Tank Controller raycasts and checks if a tank is behind a wall and therefore the shell should be on turret layer etc.) and sets the shell's z position accordingly.

Then the Shell Controller script checks its z position in the Start function, so it knows if it should go to the hull layer or not (more on that later). Z position has no effect with 2D colliders in this case, so it can be changed without causing problems with collision detection.

2 – Changing Layers

By default, all shells start from the turret layer, and move to the hull layer after the shell has traveled 2 meters after exiting the area of the tank.

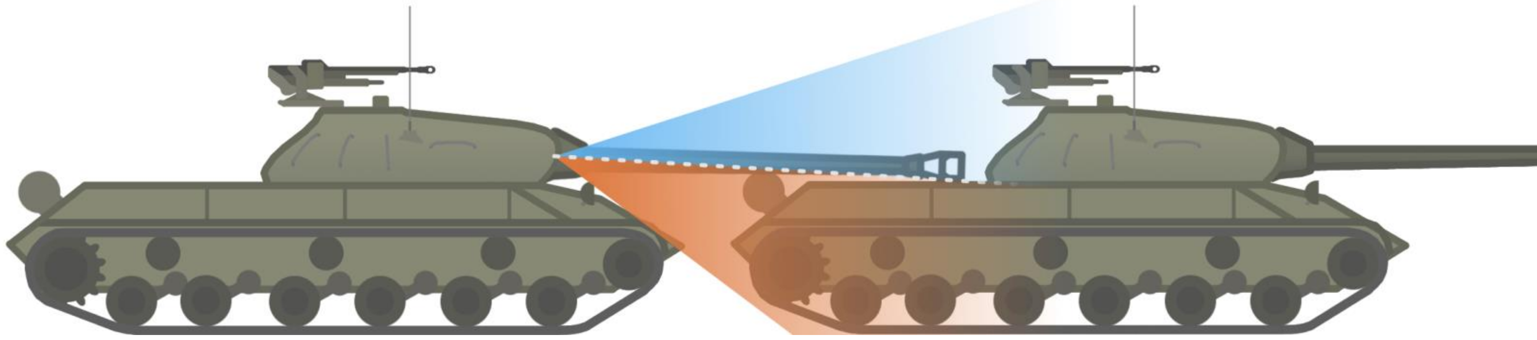
There are many reasons for this. First one is that if the shell started on the hull layer the tank would shoot itself, because in most cases, the shell "spawn point" is within the limits of the hull armor model.

Another reason is that tanks have limited gun depression, so they can only lower their guns a limited amount, usually 3-12 degrees. When two tanks are very close to each other, they realistically can't shoot each other's hulls.

It also makes "facehugging" a possible tactic, so a tank with strong turret armor can drive right up to the face of another tank and bounce incoming fire with its strong turret armor because the enemy can't shoot its hull.

This also makes shooting over tanks' engine decks possible.

This image shows how limited gun depression prevents the tank on the left from shooting the hull of the other tank.



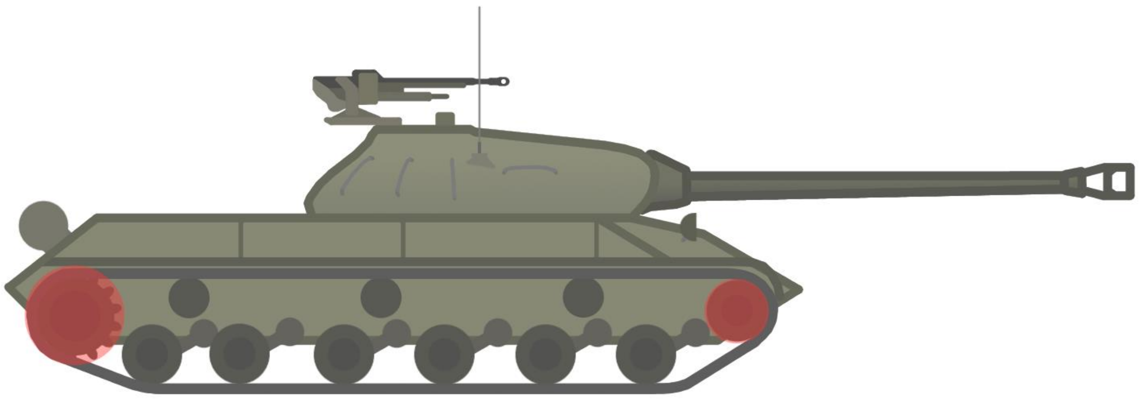
1.3 | BASICS // Shell Mechanics

3 – Hitting a Track

When a shell hits a track (doesn't matter if it's AP or HE) it first causes damage to the track piece and destroys it if (as in most cases) the damage of the shell is greater than the HP of the track.

Destroying the track in this asset package means shooting either the front or rear drive wheels, because that is the critical part of the track, and prevents the tracks from being damaged whenever any part of the side is hit.

This image shows the position of the drive wheels in a tank highlighted in red.



After the shell has hit the tracks, its penetration value will be reduced by the thickness of the track, and the penetration will be heavily reduced over time. By default, flying normally in the air with no obstructions the shell won't lose any of its penetration value.

4 – Hitting a Tank

AP SHELLS

AP shells cause damage only if they penetrate the enemy armor. Down below are the steps that are taken to see if an AP shell penetrates or not.

1. Calculating overmatch. Overmatch is a situation, where the shell hits an extremely thin armor plate, so the armor plate is unable to ricochet and deflect the shell no matter how steep the angle is. Think about a bullet hitting a sheet of tinfoil. You cannot make the bullet ricochet off the sheet of tinfoil no matter how steep of an angle you try. And that is exactly what happens with tank shells and thin armor plates.

In this asset, overmatch (and by that a penetration) will happen when the caliber of the shell is at least three times greater than the nominal thickness of the armor that it's hitting.

2. Checking for a ricochet. If no overmatch happens, the next thing to check is whether a ricochet will happen. Ricochet happens when a shell hits the armor at a very extreme angle (70 or more in this asset) and only the side of the shell hits the armor, causing the shell ricochet off the armor plate.

3. Calculating effective armor thickness. Armor plates have better effective thickness when they are placed at an angle, either by the design of the tank, or by the driver of the tank. Read more about sloped armor here: https://en.wikipedia.org/wiki/Sloped_armour.

You might wonder how constructional armor angle (e.g. sloped frontal hull plate) can be implemented with a top down 2D view that this asset has, but it is covered! The Armor Controller script, that is attached to all armor colliders, has two fields: armor thickness and armor angle. So, if you were to implement the T-55 tank into you game using this controller, you would simply type in 100 as the armor thickness, and 60 as the angle, and the Shell Controller script will calculate the effective thickness of the plate (200mm) correctly, even when the edge (or box or whatever) collider is flat on towards the shell.

4. Calculating penetration. If the shell can't overmatch the armor and the angle of impact is less than 70 degrees, penetration will be calculated. The penetration value is slightly randomized (between 0.9-1.1x), and if the randomized penetration value of the shell is more than the effective thickness of the armor a penetration will occur. The damage will be randomized between 0.9-1.1x to add slight variation.

HE SHELLS

HE shells have a very different mechanics compared to AP shells. HE shells do not have very high penetration (usually 30-80mm), so HE shells will most of the time cause damage by exploding on the surface of the armor and cause so called "splash damage". Due to this mechanic HE shells will almost always cause damage. If an HE shell manages to penetrate the armor it will cause a huge amount of damage, and therefore lightly armored tanks should avoid getting hit by HE shells at all costs.

1. Calculating overmatch. HE shells also have the very same overmatch mechanic that the AP shells have.

2. Calculating effective armor thickness. Effective armor thickness is calculated the same way as with AP shells if the shell doesn't overmatch the armor.

2. Calculating penetration. HE shells do have a similar penetration mechanic to AP shells, but their penetration value is very limited (around 30-80mm), so penetration value will be randomized the same way as with AP shells, and the randomized penetration value and the effective armor thickness are compared to check whether the shell penetrates the armor. The damage will be randomized between 0.9-1.1x to add slight variation.

3. Calculating splash damage. The main difference between the AP and HE shells is what happens when the shell fails to penetrate the armor. AP shells just bounce, but HE shells have will cause splash damage. The amount of the damage is between 0-50% of the damage stat set in the Shell Controller script, and the damage is calculated based on the ratio of the penetration value of the shell and the nominal thickness of the armor plate.

When an HE shell with 800 damage and 75 mm of penetration hits an armor plate with a thickness of 100 mm, the damage will be 300 on average, of course slightly varying based on the randomization of the penetration and damage values.

1.4 | BASICS // Scripts

There is a total of 7 scripts in this asset, and here are short descriptions of all of them:

1. Tank Controller. As the name suggests, it handles most of the tank related things.

Note that you can change all variables under the "Tank Stats" section in the Tank Controller during run time.

2. Armor Controller. Basically, two lines of code, attached to all armor pieces to give them their thickness and angle.

3. Shell Controller. Makes the shells to their job, calculate penetrations, damage tracks and inflict damage.

4. Track Mark Controller. Attached to all track mark game objects to make them fade away and last only a certain amount of time.

5. Building Destruction Controller. Attached to all destructible objects such as walls and buildings to make them vulnerable to incoming shells and tank ramming.

6. Cap Controller. Attached to the capture point to detect tanks within its area.

7. Demo Scene Controller. This script makes testing tanks easy, as you can enable and disable the input of tanks by right clicking on them and reload scene by pressing escape.

SCRIPT INTERACTION

Here is a general explanation on how all scripts interact with each other, so you can better understand how they work.

Tank Controller & Tank Controller

There are two events when a Tank Controller needs to access another Tank Controller. First event is when a UI panel is over another tank, and the script needs to know if the other tank is not destroyed, i.e. has a Tank Controller script attached to it. Other event when a Tank Controller needs to access another Tank Controller (*TankID* variable) is when two tanks collide with each other. This is required to prevent ramming calculation from happening too often with the same tank.

Tank Controller & Shell Controller

There are two events when the Tank Controller and Shell Controller scripts interact. It happens when a shell penetrates a tank or when a shell damages the tracks of a tank. In both cases, the Shell Controller script calls either *TakeDamage* or *TakeTrackDamage* function in the Tank Controller. The amount of damage is passed to the Tank Controller through the parameter of the function.

1.4 | BASICS // Scripts

Tank Controller & Building Destruction Controller

When a tank rams a building or a wall, the Building Destruction Controller accesses the Tank Controller's *PushingForce* variable to determine how fast the structure should be destroyed.

Shell Controller & Building Destruction Controller

When a shell hits a building, the Building Destruction Controller accesses the shell's damage value and subtracts the amount from its HP.

Tank Controller & Cap Controller

When the Cap Controller detects that there is only one (not destroyed) tank within its area, it increases the value of *CapPoints* variable in the Tank Controller.

Demo Scene Controller & Tank Controller

When you right click on a tank with the Demo Scene Controller script enabled, it changes the *InputEnabled* boolean in the Tank Controller to enable/disable its input.

1.5 | BASICS // Physics Layers

You can import the layers automatically when installing the asset pack, but if you already using layers 8-15 in your project, you need to create the layers manually.

Note that you should set gravity axis from Y: -9.81 to Z: -9.81 in Edit -> Project Settings -> Physics. You should also set both 2D gravity axes to 0 in Edit -> Project Settings -> Physics 2D.

There are a total of 8 physics (2D) layers in this asset package, and they are:

1. Armor - Layer for hull armor model

2. Shell - Layer for the shell on hull layer

3. Physics Model – Layer for the main GameObject of the tank with the rigidbody component

4. Armor Turret – Layer for turret armor model and the turret parent GameObject

5. Shell Turret – Layer for the shell on turret layer

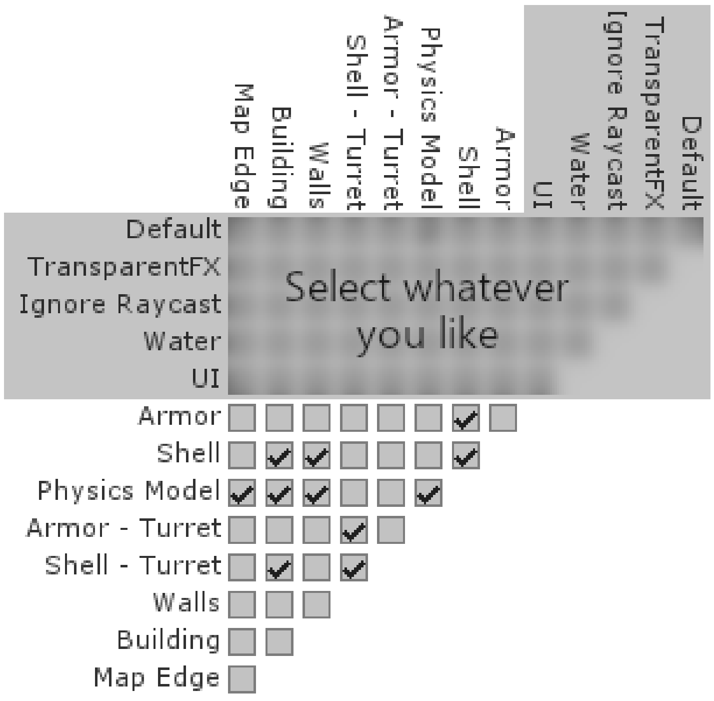
6. Walls – Layer for walls and other objects that block the shell on hull layer

7. Building – Layer for buildings and other objects that block the shell on both layers

8. Map Edge – Layer for the map edge (lets shells through but blocks the tank)

The index of the layer or the order they are in the list doesn't matter, because they are not hard coded in any of the scripts.

Here is how you should set up the collision matrix (Edit -> Project Settings -> Physics 2D):



Select whatever you like

	Map Edge	Building	Walls	Shell - Turret	Armor - Turret	Physics Model	Shell	Armor	UI	Water	Ignore Raycast	TransparentFX	Default
Default													
TransparentFX													
Ignore Raycast													
Water													
UI													
Armor	☐	☐	☐	☐	☐	☐	✓	☐					
Shell	☐	✓	✓	☐	☐	☐	✓						
Physics Model	✓	✓	✓	☐	☐	✓							
Armor - Turret	☐	☐	☐	✓	☐								
Shell - Turret	☐	✓	☐	✓									
Walls	☐	☐	☐										
Building	☐	☐											
Map Edge	☐												

1.5 | BASICS // Physics Layers

When you have set up the layers, you need to assign them to the GameObjects and scripts. Note that you only need to change the layer of those GameObjects that have a 2D collider attached to them.

GAME OBJECTS

Format: Physics Layer – GameObject(s)

Physics model – Tank parent GameObject

Armor Hull – Hull armor collision model GameObjects, Track collision model GameObjects

Armor Turret – Turret armor collision model GameObjects & turret parent GameObject

Shell Turret – Shell parent GameObject (shells start on this layer by default)

Layers Building, Walls and Map Edge are explained in the previous section and they are self-explanatory anyway.

SCRIPTS

Format: (Variable type) Variable – Physics layer

Tank Controller:

(LayerMask) hullLayerCheckMask – Walls, Building

(LayerMask) turretLayerCheckMask – Armor Turret, Building

Shell Controller:

(int) HullArmorLayerIndex – Armor Hull

(int) TurretArmorLayerIndex – Armor Turret

(int) ShellHullLayerIndex - Shell

1.6 | BASICS // Sorting Layers

Sorting layers are a lot easier to import, because they are not tied to their index, so you should just automatically import them when you're importing the asset package for the first time.

But if for some reason you're unable to automatically import sorting layers, here is the complete list of them in the right order.

"Part. ..." -prefix means that that sorting layer belongs to a particle GameObject, and the others are for GameObjects with Sprite Renderer components (and UI is for the canvas GameObject).

Background – Map background sprite

Cap Circle – Capture area background sprite

Track Marks – Track mark sprites

Track Pieces - Destroyed track piece sprites

Part. Hit Building – Particles that appear when the shell hits a building or anything that isn't a tank

Walls – Wall sprites

Part. Dust & Structure – Tank driving dust and structure destruction particles

Tank Hull – Tank hull sprite

Part. Exhaust & Shoot & Tracer – Tank exhaust smoke, shoot muzzle flash and smoke particles and shell tracer particles

Tank Turret – Tank turret sprite

Shells – Shell sprite

Buildings – Building sprites

Part. Explosion & Shell – Tank explosion and all other shell particles

UI – Sorting layer for the UI canvas GameObject

2.1 | TUTORIALS // Modifying & Creating Tanks

MODIFYING TANKS

Modifying tanks in this asset pack is very simple. All you really need to do is to play with the values in the Tank Controller and Shell Controller scripts in the editor. You can also change the armor values by changing the variables in the Armor Controller script attached to all armor model GameObjects. Hull armor model is under HullCollision parent GameObject and turret armor model can be found under Turret>TurretCollision, or under the main parent.

You can also change all variables under the "Tank Stats" section in the Tank Controller during run time.

CREATING A NEW TANK

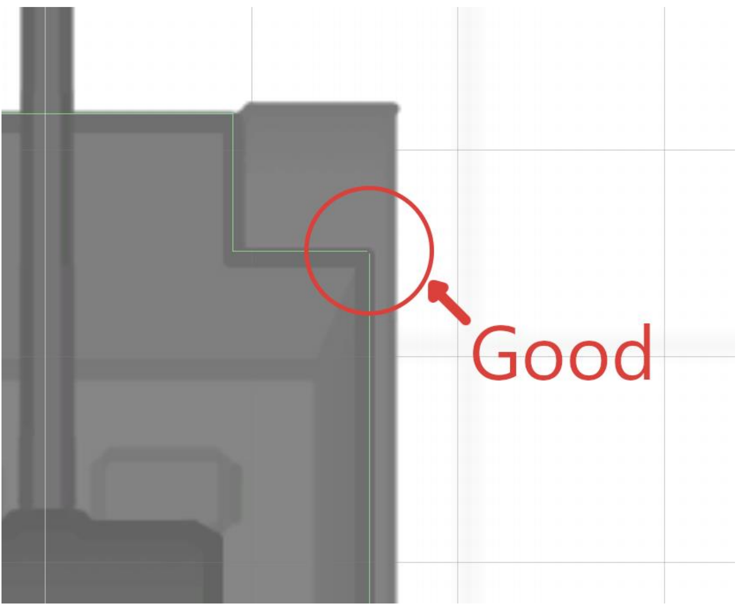
When you want to create a new tank, it is highly recommended to copy an existing tank and change its settings and components. Next up are all the needed steps to make your own tank.

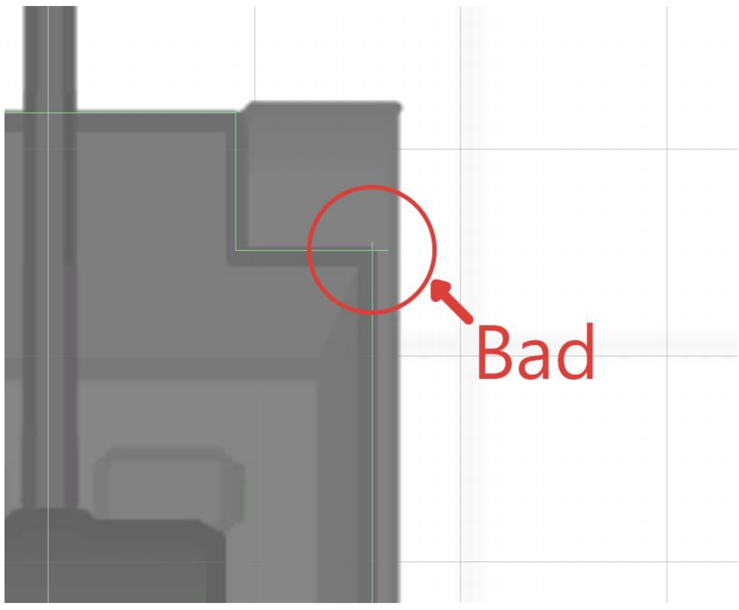
1. Make the textures. The turret and hull should have their own textures. The textures don't need to be in the middle of the image as you can fix it with the Sprite Editor in Unity later.

2. Set the main collider size. After setting up the sprites for the hull and turret you should change the size of the Box Collider in the parent GameObject of the tank so that it is the same size as the tank. This is the collider used when tanks collide with each other or with buildings.

3. Make the armor model. Then you need to create the armor model for the hull and the turret. You can make them with any of the 2D collider types, so if you want to go simple, just use box colliders, but for more detailed armor models (as used in the tanks included in the pack) you need to use Edge Colliders. Remember to set all pieces of the armor model to right layers (either to hull or turret armor layer) and attach the Armor Controller script to them and fill the *ArmorThickness* and *ArmorAngle* fields.

The collider edges do not need to line up perfectly in the corners, but it is better to have a little gap between the edges, than to have overlapping colliders.





4. Set up the tracks. Adjust the collider sizes by changing the offset and size of the Box Collider, not by changing anything in the Transform component, as it will break other things such as track mark positions and dust particles. If you have no idea how big the track colliders should be, check the other tanks for reference.

Last change the visual track piece size (child objects of each tracks) by changing the position and scale of the GameObject. This piece is shown when the track is destroyed.

5. Set up dummy objects. Then what you need to do is set up all so called dummy objects and other empty GameObjects.

Adjust the Shell Origin and Shoot Particles GameObjects under the turret. The Shell Origin (the position where the shell is spawned at) should be outside the turret armor to prevent the tank from shooting itself when firing. The Shoot Particles GameObject should be at the end of the barrel where the particle effects look the best.

Last position the UI Target and ExhaustParticlePosition GameObjects under the Dummy Objects so, that the UI panel and the exhaust particle effects look the best.

6. Finishing touches. The last thing to do before final testing and tuning is to set all stats and make the shell for the tank. Again, it is recommended to copy an existing shell, so you just need to edit the stats and maybe change some of the effects or the size of the visual model and last set the path to this new shell in the Tank Controller.

Also, don't forget to change the mass of the tank in the Rigidbody, as it has a huge effect on the mobility and ramming capabilities of the tank.

So now you have a new tank, and the final thing to do is to tweak everything to your liking.

3.1 | OTHER // Tips

Here are some other things worth noting.

Standard Assets

It is recommended to import fonts from the "Utility" part of the Standard Assets to make the UI work. You can import it from Assets -> Import Package -> Utility.

Input Axes

You find these settings in Edit -> Project Settings -> Input.

For optimal tank behavior, it is recommended to set some input axis values as shown down below. All other values can be left as they are by default.

Vertical axis (forward/backward movement)

Gravity: 10

Sensitivity: 3

Snap: enabled

Horizontal axis (rotating the tank)

Gravity: 3

Sensitivity: 3

Snap: enabled

Turret axis (only effects if you use keyboard input for turret)

Gravity: 3

Sensitivity: 3

Snap: enabled

Fire axis

Gravity: 10

Sensitivity: 3

Snap: disable

3.2 | OTHER // Changelog

When the pack or the documentation is updated the changes will appear here.

If you're reading the offline version of this document, it is recommended to use the online version, since the offline version provided with the pack is updated only when the pack updates.

You can find the link to the updated version on the Asset Store page and on my Google+ page.

My Google+ page: https://plus.google.com/u/0/106742955221032886735