TABLE OF CONTENTS

WELCOME TO DEFTLY

Thanks for purchasing Deftly by Cleverous!
This is a basic Getting Started document. It contains information to build the fundamental prefabs.

BEFORE YOU GET STARTED

The way Deftly works is it gives you a few very open-ended scripts that you are required to populate with data. It handles all of the heavy lifting in the background and does so in a fashionably dependable and expedient manner.

No scripting is required to use the features that Deftly offers.

You should expect to get out of Deftly what you put in. As the game developer in this equation you will be responsible for providing all of the models, effects, graphics, sounds, animations and other trinkets that you would like to see in your game and Deftly will breathe life into it.

You can expect a good front end for designing cool features, such as:

- Characters can be designed with the Subject script.
- Weapons can be designed with the Weapon and Projectile script.
- Controls are exposed by the Deftly Camera and PlayerController scripts.
- AI can be designed with the Intellect script.
- Future updates will give you even more options, more control and user-requested features.

The Demo scene provides examples for all of that. Please take a look at it and study the example prefabs.

DONT STRUGGLE!

If you are struggling with some feature, found a bug, hate something in Deftly or anything like that then please respond to the official thread on the Unity forums or contact me directly by one of the ways outlined in the Contact section so we can make Deftly work for you.

This is an early release product and is a great time to carve in the features YOU want to see in it, so make yourself heard!

Official Website: **http://www.cleverous.com/**
Online Documentation: **http://www.cleverous.com/#!tds-getting-started/cht7**
Release Notes: **http://www.cleverous.com/#!Deftly-release-notes/c1cdb**
Trello Roadmap: **https://trello.com/b/Wc9Zqt6r/cleverous-tds-kit-Deftly**
Live Chat Channel: **https://jabbr.net/#/rooms/Cleverous**
Email: **lanefox@gmail.com**

Use Deftly Global Options to customize a variety of options like
Floating Text, Weapon AutoSwapping and more
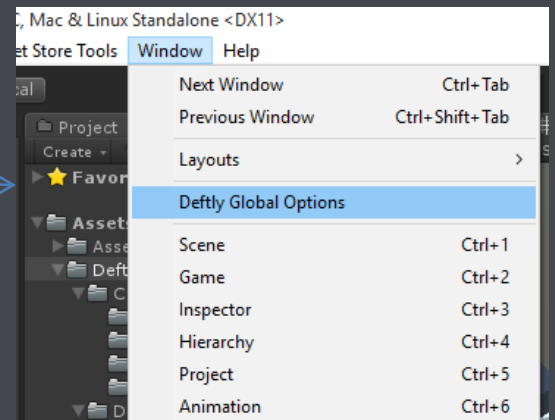
Setup the Camera because it is the all powerful game overlord

I. On your Main Camera (which should be tagged as Main Camera) add a Deftly Camera script
II. Increase "Targets" to 1
III. Drag your character from the Scene into the Targets slot
IV. Changes to the actual transform of the camera now do nothing. They are controlled by the script.
V. Instead, modify the Offset parameters. This is the distance in each direction from the Target Position
VI. The "Target Position" is the *average position of all targets* so if you have more than one target it will be looking at the average weighted position between them.
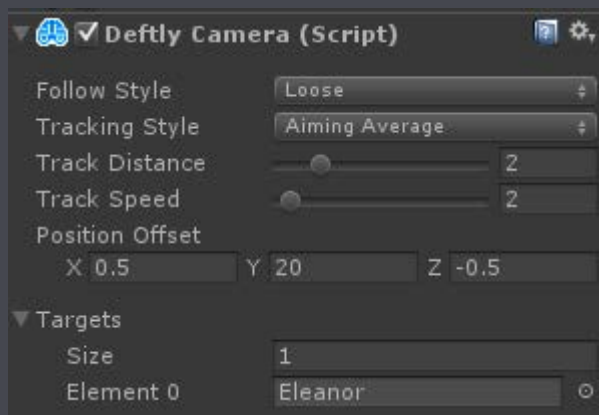
Regarding this camera script, it is very very early in design! Currently it has a few bugs and the inspector is uninviting. Sorry about that, we'll get it whipped into shape soon. If it starts acting up, just kick it until it does what you want.

Soon it will support live realtime updates of players joining/exiting the game and be more intuitive to use.

All changes in the editor will carry over to runtime while runtime changes do not carry back to editor mode. You can find more information on the Trello Roadmap.

Eventually, it seems like the Camera is the ideal place to manage the game so you might start seeing this area get a lot more feature options by default.

If it dies, its a Subject
Currently requires *that you are using a configured Deftly Camera*

To create a Player you need a rigged and Mecanim-ready Humanoid character model.

I.   Create an Empty GameObject (this is the "Container" object)
II.   Place your Character Model as a Child of it, adjust as needed.
III. On its Animator component choose an Animator Controller and disable Root Motion.
IV.   Back to the Container Object... Add a Capsule Collider and scale it to fit.
V.   Add a PlayerController script. (This forcefully adds a Subject and Rigidbody script for you.)
VI.   Done with adding components
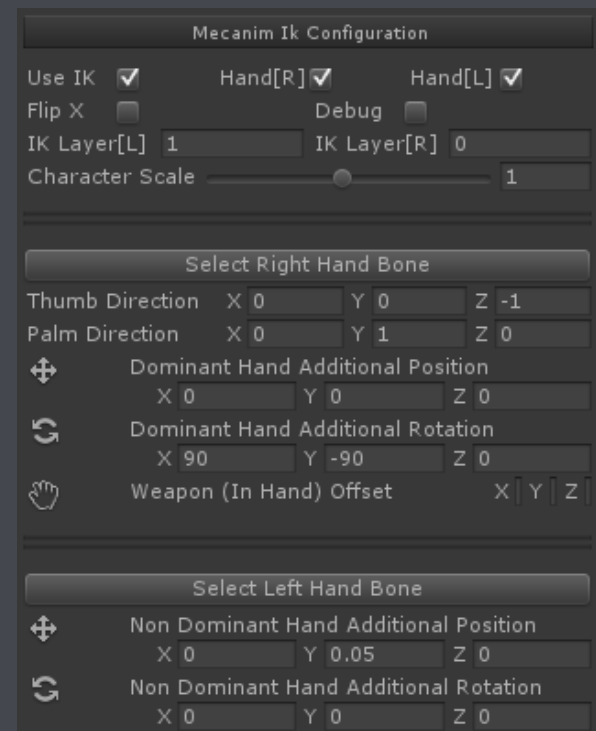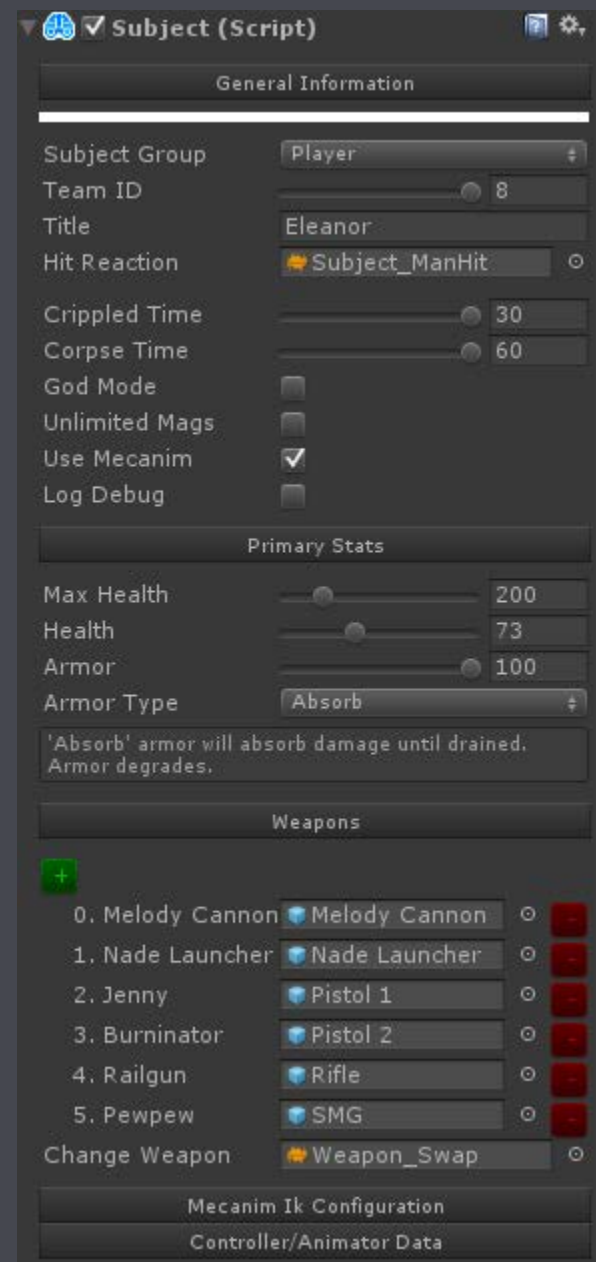

NOW WE NEED TO CUSTOMIZE THIS SUBJECT

The majority of default values are ok, but we'll need to specify some settings that are unique to your project, like layers, masks, etc.

I.   First tell the Subject where the Animator is. Drag your model into the Animator Host Obj field.
II.   On the Player Controller script, specify the Mask field (all layers the cursor can aim to, usually just "Floor, Environment" or something.)
III. Specify your Player as one of the Targets in the Inspector.
IV.   Now you can press play and control your character. (if you have a camera)
V.   Make it a prefab (Drag to Project Hierarchy)
VI.

        YouTube Tutorial for setting up IK features

POSSIBLE PROBLEMS

- Animator Host Obj is undefined/incorrect
- Animator Controller was not specified on the Model's Animator.
- Floor is not actually on the Floor Layer (check Mask in PlayerController) or missing a collider
- Player is below the floor or under/inside a collider
- Health starts at zero.
- Character Model import settings not set to "Humanoid" or Root Motion (Animator component) is enabled.
- Subject toggle option called "Use Mecanim" was turned off

▼ 🧠 ☑ Subject (Script)   🔲 ⚙,

General Information

Subject Group | Player   ⬍
Team ID | ⊙ 8
Title | Eleanor
Hit Reaction | 📁 Subject_ManHit   ⊙

Crippled Time | ⊙ 30
Corpse Time | ⊙ 60
God Mode | ☐
Unlimited Mags | ☐
Use Mecanim | ☑
Log Debug | ☐

Primary Stats

Max Health |───●───────────| 200
Health |──────────●────| 73
Armor |─────────────●─| 100
Armor Type | Absorb   ⬍
'Absorb' armor will absorb damage until drained.
Armor degrades.

Weapons

➕

0. Melody Cannon 📘 Melody Cannon   ⊙ —
1. Nade Launcher   📘 Nade Launcher   ⊙ —
2. Jenny   📗 Pistol 1   ⊙ —
3. Burninator   📗 Pistol 2   ⊙ —
4. Railgun   📗 Rifle   ⊙ —
5. Pewpew   📗 SMG   ⊙ —
Change Weapon   📁 Weapon_Swap   ⊙

Mecanim Ik Configuration
Controller/Animator Data

Mecanim Ik Configuration

Use IK ☑   Hand[R] ☑   Hand[L] ☑
Flip X ☐   Debug ☐
IK Layer[L] 1   IK Layer[R] 0
Character Scale |──────●──────| 1

Select Right Hand Bone

Thumb Direction   X 0   Y 0   Z -1
Palm Direction   X 0   Y 1   Z 0
✛   Dominant Hand Additional Position
          X 0   Y 0   Z 0
🔄   Dominant Hand Additional Rotation
          X 90   Y -90   Z 0
✋   Weapon (In Hand) Offset   X Y Z

Select Left Hand Bone

✛   Non Dominant Hand Additional Position
          X 0   Y 0.05   Z 0
🔄   Non Dominant Hand Additional Rotation
          X 0   Y 0   Z 0

HOW TO MAKE A RANGED WEAPON

These go pew pew

I.   Make an Empty GameObject (container object)
II. Add your model as a Child of it.
III. On the container object add a Weapon script

You'll see a couple of red errors in the Inspector. First, you must have a Spawn Point for the Projectile to be created at. The other is regarding IK, you cannot have a Weapon origin at 0,0,0, there must be some offset from the bone it is linked to.

Be mindful of the position

... that you choose for the model as a child of the wrapper.
The Weapon GameObject's will be put as a child of the Hand Bone of the Subject that owns it and zero'd out. This means that the pivot of the Weapon prefab will be exactly on the pivot of the Hand Bone. You can specify additional offset, for instance a small offset to move it into the palm better is basically standard practice here.
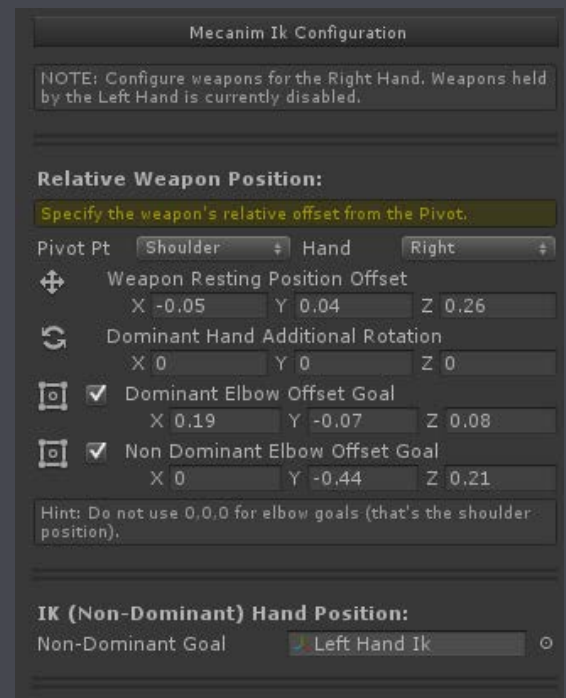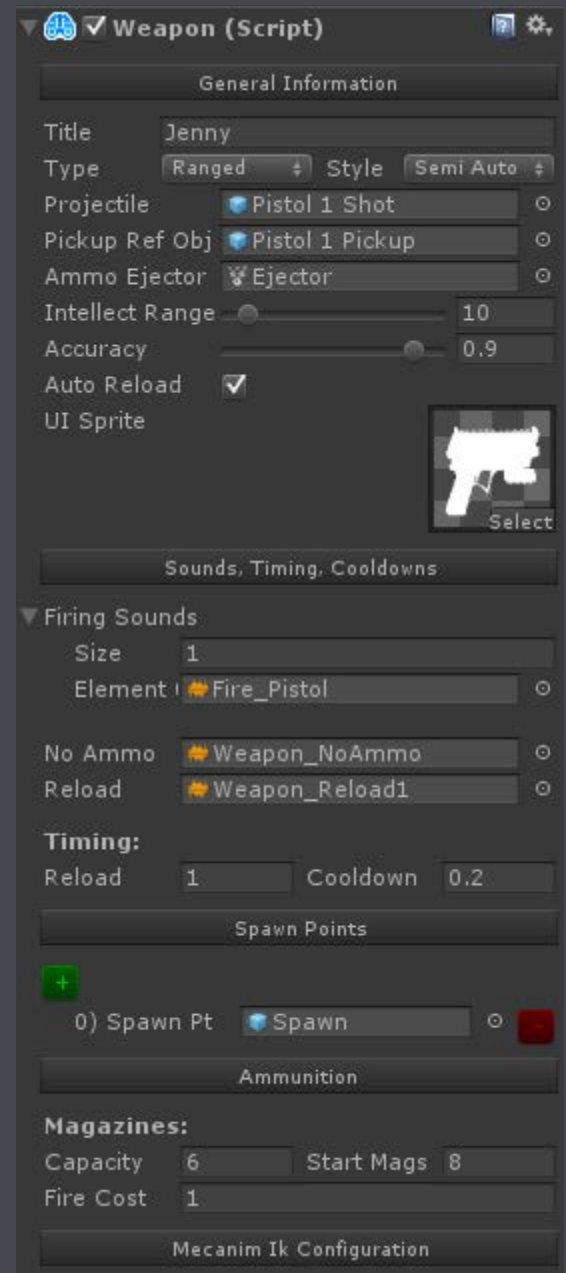
I.   Customize and populate the basic fields (Audio clips, accuracy, title, magazines, etc..)
II. Create an empty GameObject, make it a child, place it near the barrel.
III. Under "Projectile Spawn Points" press the [+] button and put your spawn point in the new field.
IV. Create another empty GameObject to be used as a target for the Left Hand.
V.   Choose the "Pivot From" and input an offset from which this gameobject will stand off from that bone.

Now lets setup the IK stuff...
1. Weapon Resting Position Offset
   a. Position tweaking for Weapon relative to Hand
2. Dominant Hand Additional Rotation
   a. not working yet =(... Next Patch.
3. Dominant / Non Dominant Elbow Goals
   a. Positional tweaks for the elbow positions.
4. Non Dominant Hand Goal Object
   a. An object representing pos/rot goal for the Left Hand

We now have a Weapon, but no Projectiles for it to fire! We need a prefab to put in "Projectile" field in the upper right of the inspector.

Make this Weapon a prefab. (Drag into Project Hierarchy)

☑ Weapon (Script)

General Information

Title	Jenny
Type	Ranged Style Semi Auto
Projectile	Pistol 1 Shot
Pickup Ref Obj	Pistol 1 Pickup
Ammo Ejector	Ejector
Intellect Range	10
Accuracy	0.9
Auto Reload	☑
UI Sprite	Select

Sounds, Timing, Cooldowns

▼ Firing Sounds
  Size   1
  Element Fire_Pistol

No Ammo   Weapon_NoAmmo
Reload   Weapon_Reload1

Timing:
Reload 1   Cooldown 0.2

Spawn Points

0) Spawn Pt   Spawn

Ammunition

Magazines:
Capacity 6   Start Mags 8
Fire Cost 1

Mecanim Ik Configuration

Mecanim Ik Configuration

NOTE: Configure weapons for the Right Hand. Weapons held by the Left Hand is currently disabled.

Relative Weapon Position:
Specify the weapon's relative offset from the Pivot.

Pivot Pt Shoulder ♦ Hand   Right   ♦

⊕   Weapon Resting Position Offset
      X -0.05   Y 0.04   Z 0.26
⟳   Dominant Hand Additional Rotation
      X 0   Y 0   Z 0
▣ ☑ Dominant Elbow Offset Goal
      X 0.19   Y -0.07   Z 0.08
▣ ☑ Non Dominant Elbow Offset Goal
      X 0   Y -0.44   Z 0.21

Hint: Do not use 0,0,0 for elbow goals (that's the shoulder position).

IK (Non-Dominant) Hand Position:
Non-Dominant Goal   Left Hand Ik

Similar to making a Weapon.

I.   Make an empty GameObject.
II. Add a Projectile script to it.
III. Configure basic parameters (damage, speed, etc)
IV. Add a rigidbody and disable Gravity. You could also lock rotation on all axis.
V.   Add a Collider

Thats all there really is to it, you can refer to the tooltips on fields to better understand what each one does.

However, I'll briefly explain the tags system.

Hit [+] to add a tag filter. Notice there is Tag Name, Audio Clip and GameObject fields. You'll type in the name of the Tag and specify the corresponding effects you want it to use when it hits something on that tag.

For instance "Wood" for the name, "Wood_Impact" audioclip and "Breaking Wood Particles" prefab as the gameobject field. You'll change the tag of the wooden game objects to Wood and the Projectiles will react accordingly. The first tag is the fallback effect if something unrecognized is hit.
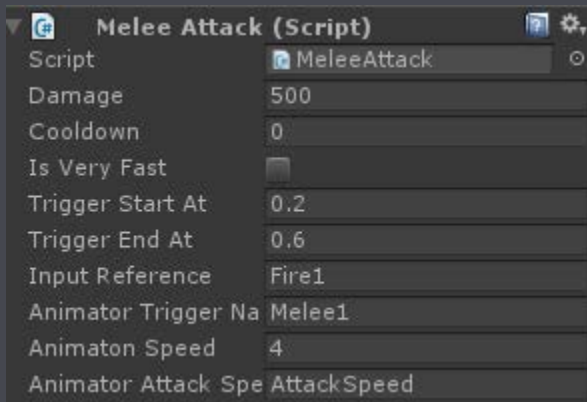
NOTE: It's possible that we'll move to a global control panel for these Impact Tag groups and you'll just identify which category each projectile will inherit from... Communicate if you have concerns about this possible change.

Make this a Prefab (Drag into Project Hierarchy).
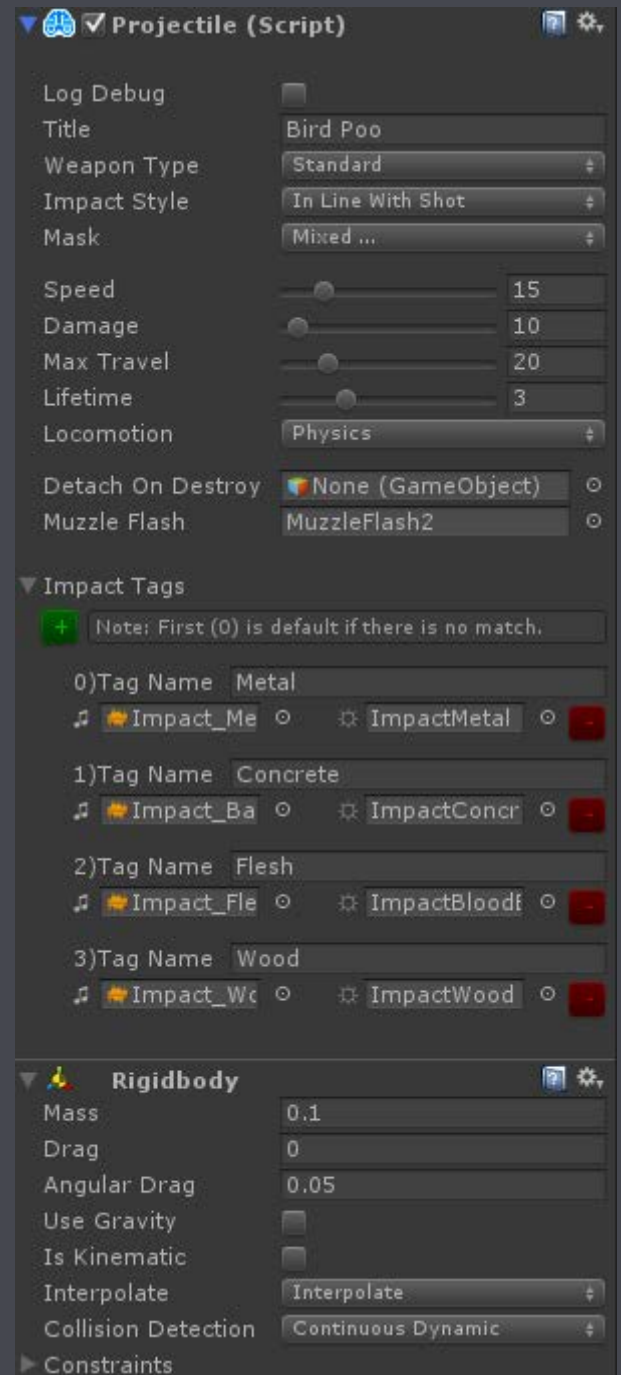
HOW TO MAKE A MELEE WEAPON

Melee Weapons are super unpolished and prone to glitches right now

Melee Attack (Script)

Script	MeleeAttack
Damage	500
Cooldown	0
Is Very Fast	☐
Trigger Start At	0.2
Trigger End At	0.6
Input Reference	Fire1
Animator Trigger Na	Melee1
Animaton Speed	4
Animator Attack Spe	AttackSpeed

First, get acquainted with how Ranged Weapons work and then take a whirl at Melee Weapons. They're not complete but do work in function. They operate differently in that they require instead of a Projectile, they use a Melee Attack. The Melee Attack inspector is not custom yet and some features are not enabled.
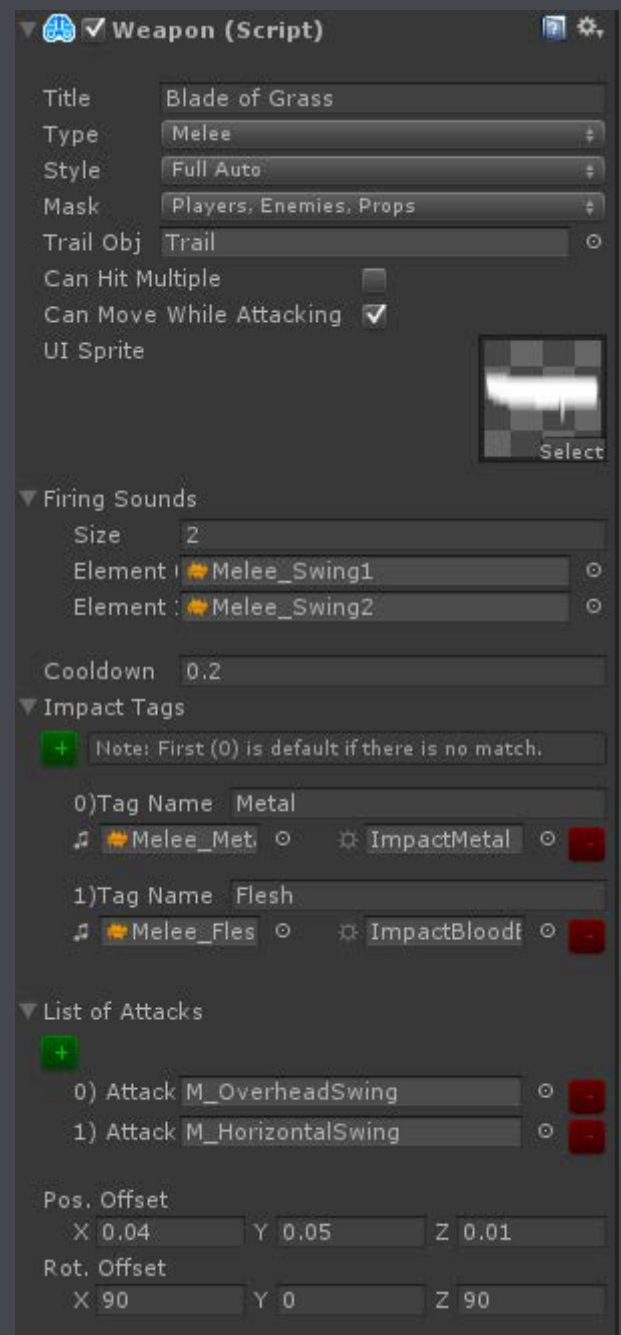
First change the Type from Ranged to Melee.

The Melee Weapons use a Trigger instead of Collider, when the trigger passes through a collider it will try to damage it. In order to prevent the Weapon from 'touching' things (for example swiging the Weapon back to prepare for a big forward attack) and damaging unintentionally you should specify a smaller time window for the Trigger to be enabled.

When the animation plays, the Melee Attack will turn on/off based on how complete that animation is from zero to one (start to end).

In the future we'll make some alternative ways to do this and improve the polish on Melee Weapons.

- **Cooldown**: How long after an attack before you can attack again.
- **Trigger Start At**: Percentage of animation where the trigger will turn on.
- **Trigger End At**: Percentage of animation where the trigger will turn off.
- **Animator Trigger Name**: What Trigger you want to send to the Animator Component. Your Animator Controller needs to be configured to accept these inputs and fire the correct trigger.
- **Animation Speed**: The numerical playback speed multipler.
- **Animator Attack Speed**: The name of the variable that controls the animation's playback speed. This should be linked in the animation's state to this variable. It's a fairly new feature (5.1.2 I think), so you probably aren't aquainted with it just yet.

Once you make your Melee Attack a Prefab then you can throw it into the Weapon script's List of Attacks category . Refer to the demo "Blade Of Grass" and its Melee Attack's for a working example.

Weapon (Script)

Title	Blade of Grass
Type	Melee
Style	Full Auto
Mask	Players, Enemies, Props
Trail Obj	Trail
Can Hit Multiple	☐
Can Move While Attacking	☑
UI Sprite	

Firing Sounds

Size	2
Element (	Melee_Swing1
Element :	Melee_Swing2

Cooldown	0.2

Impact Tags

Note: First (0) is default if there is no match.

0)Tag Name	Metal
♪ Melee_Met	☼ ImpactMetal

1)Tag Name	Flesh
♪ Melee_Fles	☼ ImpactBloodE

List of Attacks

0) Attack	M_OverheadSwing
1) Attack	M_HorizontalSwing

Pos. Offset

X 0.04	Y 0.05	Z 0.01

Rot. Offset

X 90	Y 0	Z 90

HOW TO WRAP IT UP

Put all the pieces together!

Now you should have a Prefab of a Subject, Weapon and Projectile.

Select the Weapon Prefab and put the Projectile Prefab in the "Projectile" slot for the Weapon. Now when the Weapon fires it will be spawning that prefab. Pooling will be in a future release.

Now select the Subject Prefab and press the + button to add a new Weapon Slot. Fill that slot with the Weapon Prefab. Now when that Subject is brought into the scene at runtime it will be using that first Weapon slot. Do not leave empty slots.

Easy peasy!

CONTROLLER SUPPORT VIA INCONTROL (BY GALLANT GAMES)

If you want to use Controller's in Unity and do it properly you will be using a plugin like InControl, cInput or ReWired. Currently there is only very basic Controller support built into Unity by default and a more robust system is necessary if you plan to handle various controller types, cross-platform development and just ease of life improvements in general.

I.   Import InControl to the Project. You might need it.
II.   Navigate to Deftly / Core / Addons / and open InControl Support.zip
III.   Replace the existing PlayerController script in the project with the one in the zip file.
IV.   Add InControl to the scene by GameObject > Create Other > InControl > Manager
V.   On the PlayerController script that will be using a Controller, toggle '*Use In Control*' to ON.
VI.   Device ID can be specified. If you only have one device connected then it will always be zero.

If necessary there is an original script included in a zip file as well. You can revert with that if you ever need to strip InControl support.

NOTE: This was tested with a PS4 controller only for this intial support release. If you're using a different controller please double check there are no FAQ answers or 'gotchas' in the InControl documentation that may resolve the problem. Report issues you can't resolve and I will look into fixing them.

FLOATING TEXT AND DAMAGE

Why not both?

Floating Damage is a three part system.

1. Global Option to enable Floating Text
2. Prefab in a Resources folder which is used for all Floating Text
3. GUIText and Floating Text components on that prefab

To use the Floating Text system, go to Window > Deftly Global Options and toggle on the Show Floating Damage checkbox if it is not already on. Doing this makes every Subject script spawn a Floating Text when it processes damage. It is required that you have the bulletpoints above completed.

Confirm that...

1. Floating Text is enabled, the prefab name is correct and the Options Window has green text saying everything is okay.
2. The prefab is correctly configured: Need GUIText and Floating Text components. Refer to example in Demo/Prefabs/Resources for a goby, its pretty straightforward.
3. The prefab is in a Resources folder. This is required, it uses Resources.Load() to grab the prefab when the game starts so if the prefab is not in a folder called Resources then Unity cannot find it. Ever.

Once the prefab is spawned the Floating Text script will position itself and start floating away based on the variables you have put in those fields to configure it.

That is all that is required to make Floating Text work. *Again*, you must enable Show Floating Damage and type in the name of your Prefab that must also have a GUIText component and Floating Text component on it. Expect epic failure otherwise.

If its not working properly, try changing the text size or removing the material. You can always reference the demo scene for a proper example.

POWERUPS

And those sparkly bits

Powerups are easy. =)

1. Make a Prefab with a Trigger Collider
2. Add a Powerup component to it
3. Configure the Powerup to do what you want (this is quite straightforward, but use the tooltips if necessary)
4. Profit

You can implement their collision detection however you want, but I made the demo Powerups by making a Collider (to keep it off of the ground), a Trigger (for the Pickup detection) and a Rigidbody. The layers of these object can be independent from each other – ie the Trigger could be on some layer, then the Collider on another while the Collision Matrix configuration handles your unique scenario (if your design requires it).

You aren't forced to do it that way, but a Powerup requires a Trigger.

More features for Powerup stuff soon.

WEAPON PICKUPS

Not your average powerups

These aren't like Powerups, they're intentionally separate and you have to use the built in Pickup Reference field on each Weapon to tell it what prefab is designated as its Pickup.

The reason is because when a Weapon is dropped it will have to actually drop something and it can't be the same object because that Weapon object that the Character is carrying is not designed for that purpose.

So, we'll just make a simpler object to float around as loot and it'll be way easier to manage and design around.

I. Create a prefab with a Trigger, Collider and Rigidbody.
II. Add a Weapon Pickup script
III. Reference the Weapon Prefab which it will give, customize any fields.
IV. Add the model you want to show
V. Adjust any trigger and collider sizes and you're good to go!

You can always check out the demo for example prefabs to see how they are intended to work.

Well, we've gotta shoot something..

Spawners are handled by a Spawn Trigger, when something on the specified layer enters the trigger it will create the specified Spawners in the specified positions. If you have two positions, you must specify two Spawners because it looks to the 2nd entry when it considers the 2nd position. You'll get errors if you don't do this. Custom inspector in next patch to make this easier.

Spawners can be sort of 'invisible wave spawners' (*Fixed Count*) or 'physical spawner buildings' (*Destroyable*) that you must destroy to stop. If you want an invisible wave spawner, just make an empty GameObject and add a Spawner Script – save as prefab and you're ready to go. If you want an interactive Spawner you will need a Subject script on it (which means Rigidbody and Collider too!) so that it can properly recieve damage and die.

Destroyable Spawner
I. Create and Empty GameObject
II. Set Layer and Tag properly (it is an enemy)
III. Add a Subject Script
IV. Choose Subject Group as "Other"
V. Specify Health in Stats
VI. Add a Spawner Script
VII. Choose Type of 'Destroyable'
VIII. Add at least one Enemy Prefab and specify how many to spawn per wave
IX. Choose Spawn Points (Empty GameObjects as positions)
X. Save as prefab.

This Spawner will now Spawn the indicated enemies and their amounts, randomly at any of the provided positions, in waves, until you kill it. However now you probably need a Spawn Trigger if you're planning on popping this in a level when you enter a room or something.

Spawn Trigger
I. Create an Empty Game Object
II. Add a Box Collider and toggle Is Trigger to on and adjust the size to what you want.
III. Add a Spawn Trigger script
IV. Specify the Positions (other empty GameObjects)
V. Specify which Prefabs of Spawners you want in those positions
VI. If you want the Spawn Trigger to manage door locking when triggered – specify which doors will be affected. Those doors should probably have their "Manually Triggered" bool turned on, although Locked will be toggled by the Spawn Trigger anyway so they can't be opened until the Spawn Trigger says the victory conditions are met.
VII. Victory conditions are: All Spawners and enemies spawned are dead. If you're using a Fixed Count Spawner it will die by itself when the last wave spawns. Destroyable Spawners will continue spawning waves until you beat some sense into them.

MISCELLANEOUS

LIMITATIONS

Stuff you can't do.

- Varied floor/terrain height. (actually, this sort of works, but its not really supported very well yet)
- Use empty Weapon slots
- ...

QUESTIONS?

Personally written letters by mail are preferred

Please contact me by any of the methods outlined in the contact section on the first page. Deftly is in early development stages and will benefit greatly from your input! Help shape it by telling us how we can improve it and by reporting bugs when you find them. =)