Mann The Barricades!

Buildable objects for MVM

By LeSwordfish
With thanks to A Boojum Snark, Hydrogen and Benoist

These hologram whatsits turn into real things if you punch them hard enough. Use them to fight robots better. As we speak I am wrestling a leopard.
- Mann Co. CEO Saxton Hale, entirety of Holographic Defence Assets press release.

Accreditation

If you wish to use this in your map, i would appreciate that you mention my username (LeSwordfish) in your map page or readme, and include a link to wherever you got this from.

My thanks to A Boojum Snark, who put together the Sawblade prefab, and to Benoist and Hydrogen who helped me with the Navigation stuff.

How it Works: Player's Perspective

1. In between waves, holographic props appear, along with instructions for their use.
2. Attack the props to spend "build points" on them- when you have purchased a hologram it turns into a full prop.
3. Sprites next to props mark their cost, with props with a greater gameplay effect costing more.
4. Like spending cash, if you pass the wave your choices are consolidated: if you fail you get to choose again.
5. You earn build points however the mapper wants you to. It should be at least a few per wave, though, and I would suggest giving out more for performing certain feats (killing a tank in ten seconds, airblasting three bots to death in one wave, whatever).

How it Works: Mapper's Perspective

The whole thing is too complex to go into here- further details are given in the vmfs, and instructions on usage are below. For now, suffice to say you need to place the "Common" prefab in your level, along with as many or as few others as you feel you need. You can then plug the start and end of your waves into the Common prefab (see the instructions below), customise the buildables you place, and it should all work.

Some quick notes on the technology:
1. Holograms are prop_dynamics that are enabled and disabled,
2. Math_counters are used to track points earned and spent,

3. Game_texts display information on screen (game_text_tf is prettier but only appears outside minmode HUD)
4. Lots of logic_relays enable and disable each other so that the objects appear and disappear as needed.
5. A trigger_multiple is also used, TouchTest-ing itself to prevent players getting stuck when the holograms reenable.

Included in This Pack

The pack contains a set of prefabs - put them wherever the rest of your prefabs are and add them to the level as normal. (Why not instances? Because instances are harder to customise (and because I never could get the hang of sending inputs to them)).

buildables_common

- Sets up the necessary background for the buildables to work.
- Includes relays and logic entities to run the "background" of the buildables, game_texts for the "HUD" and an ambient_generic to play a sound when a point is spent.

buildables_prop

- Contains a basic prop that can be built and the secondary entities required for it.

buildables_break_prop

- Contains two basic props, one of which is removed when the other is built. Useful for breaking bridges, opening doors, etc.

buildables_pumpkin

- Contains a single respawning pumpkin bomb.
- Pumpkin bombs attempt to respawn every twenty seconds.

buildables_health_ammo

- Contains a single respawning large health and a single respawning large ammo pack

buildables_dispenser

- Contains a single level three red dispenser.
- This dispenser has 800 health (represented by a wrangler-like shield) and if destroyed will respawn at the end of a wave.

buildables_sentry

- Contains a single respawning level three red sentry
- This sentry has 800 health (represented by a wrangler-like shield) and if destroyed will respawn at the end of a wave.
- It is both sappable and repairable.

buildables_spell

- Contains a single respawning spellbook.

- Listen, I don’t know how to make spellbooks work. This allows you to build and spawn one, but everything else is up to you. Have fun.

buildables_sawblade
- It’s a circular saw blade like those on Sawmill.

buildables_glow
- Contains the entities needed to make a single buildable glow so it can be seen through walls.

buildables_dynamic_navblock
- Contains a sample method to dynamically block/unblock navmeshes (so you can break bridges, close doors and block/open routes for bots.)
- This is tricky to use - read the instructions below!

Instructions - Basic

1) Set up the "Common" prefab
- Place the "buildables_common" prefab in your map.
- If you are NOT using the standard mvm setup from the example vmf, you will need to connect the wave relays.
    - Send "Trigger" inputs from your wave start relay to build_wave_start and from your wave finished relay to build_wave_passed.
    - Send a "trigger" input to build_wave_failed from anything that only triggers when the wave is failed, like the logic_relay triggered by bomb deployment. Make sure it is fired both by tanks and bots, though.
    - If you are using the mvm_example vmf setup, this should be done automatically (by duplicating the example entities.)
- Set up conditions for players to earn build points, and have them send FireUserX inputs to build_point_added.
    - FireUser1 adds 1 point, FireUser2 adds 2 points, etc etc.
    - Of course, to send more than 4 points, just fire more than one input.
- Set the starting amount of build points in the logic_auto (by default it is four, you may want to adjust this)
- If needed, change the color of the game_texts for maximum visibility in your map.

2) Place and set up the "prop" prefabs.
- Place the buildable_prop/buildable_break_prop prefab.
- Customise the prop_dynamics: change the props, add more, whatever. So long as the names remain the same, the props will behave the same way.
    - For the breaking props, the props with "remove" in the name are those that will be removed when the buildable is built.

- Check the build_touchtest will prevent players from getting stuck in your props when the collision reactivates.
- Change the cost of the props, by adjusting the max value of the prop's Math_counter.
    - If this is more than six, you will need to add more sprites and more cases in the logic_case.
    - If it is less, you do not NEED to delete any spare sprites: doing this will be considerably neater though.
    - Remember to clear up any broken outputs.
- To make props glow or block paths, see Instructions - Advanced below.
- Anything you wish to be "purchased" with the buildable should be enabled in the build_create relay and disabled in the waveend_replace relay.
    - These two relays control everything for this buildable. So long as it's turned on by build_create and off by waveend_replace, go nuts.

3) Repeat stage 2 for any other props you want to place.

- I would RECOMMEND keeping buildable props simple and only having a few- 8-10 at most.
- If your map can take it, however, feel free to have more.

4) Place any other buildables.

- Place as many of the buildable_sentry, buildable_dispenser, buildable_pumpkin and buildable_item props as you want.
    - Unless you want to change the price of these, they should need no other work. (The price can be changed the same way as for other buildables)

If you have any trouble, contact me at http://steamcommunity.com/profiles/76561198010698558/ or @LeSwordfish on either tf2maps or twitter.

Instructions - Advanced

Customising Buildables

Basically, the sky's the limit. Anything that can be enabled/disabled can be controlled through the build_create and waveend_replace relays. Go wild.

Glowing Props

- If you want an object to glow (so it can be seen through walls like the Payload cart, place a buildables_glow prefab for each buildable_ entity.
- Parent each prop to a separate func_tracktrain. This will eat up entity counts fast, so is not recommended.
- Go easy on this! These buildables are a bulky system already, and making everything glow just magnifies that. I wouldn't do it myself, but the option is there.

Altering the Navmesh

Sometimes you might want to dynamically adjust the navigation mesh for the level. This is a bit more complex than can be set up with these prefabs so you'll need to do a little legwork yourself.
I'm going to assume for a moment that you know the basics of editing a navmesh. My sincere thanks to Benoist, who walked me through this, and I guess I should shoutout to valve - they used this method in Rottenburg.

- Place the buildables_dynamic_navblock prefab, or create your own - it's just a func_door. Place it so when it is closed, it blocks the area you want bots not to path through.
    - Unlike a func_nav_avoid, you don't have to cover a whole nav area (in fact, the opposite!)
    - It only needs to block the navmesh (usually on the ground level).
- When you want it to block nav, send it a Close input so it pops up out of the ground. (For an ordinary barrier, send this from the relevant buildable's build_create relay.)
- When you want it to unblock nav, send it an Open input so it goes back into the ground. (For an ordinary barrier, send this from the relevant buildable's waveend_replace relay.)
- When you've generated the navmesh, go to each navigation area you want this prop to block/unblock. For each area, point at it in nav_edit mode and in the console type "tf_mark DOOR_ALWAYS_BLOCKS" (without quotes, of course.)
- The buildables_common prefab includes a tf_point_nav_interface and a logic_relay that causes it to rebuild the nav paths several times. This automatically runs at the start and end of each wave: if you want it to run at any other time, send a Trigger input to the logic_relay.
    - (For example, if you want a wall like Rottenburg that can be destroyed, trigger it after the wall has broken.)

Any nav areas that you've marked with DOOR_ALWAYS_BLOCKS will, when intersecting a func_door, be updated and marked as Blocked when the point_nav_interface is triggered. This means bots will never path through them (happily taking func_nav_avoid routes instead.) However, the whole area will be marked, no matter how little intersects the func_door! This could lead to weird results, such as whole rooms becoming impassable because one path out of them is blocked. Pay attention to which areas are marked as Blocked when running around with nav_edit on, and be willing to split large nav areas into smaller sections.

To create new routes that the bots can take, you'll need to run nav_generate when the new route is open (or create a new nav area manually.) You can then simply set the func_door up the other way around - it is Closed (and blocking the route) when the prop doesn't exist/is a hologram) and is Open (allowing the route) when the prop has been built.

Notes

- The maximum number of build points a team can have at any one time is 15.
    - If you want to increase this, you can set up the second logic_case and the dozens of game_texts yourself, frankly.

- Pumpkins attempt to respawn every twenty seconds.
- Exploding pumpkins may damage other buildables, spending points.
    - To avoid this, add a "filter_damage_type" and set it to "blast"