

Pickups

Pickup

For non-respawning pickups, only use the idProp2/idMover entities.

idProp2

Remove the **bindInfo** if this is a self-respawning pickup.

```
entity {
  entityDef deathmatch_pickup_mega_health_1 {
    inherit = "pickup/health/large";
    class = "idProp2";
    expandInheritance = false;
    poolCount = 0;
    poolGranularity = 2;
    networkReplicated = true; // Must be true
    disableAIPooling = false;
    edit = {
      bindInfo = { // remove if self-respawning pickup
        bindParent = "deathmatch_pickup_mega_health_1_offset_mover"; // fixes entity offset
      }
      renderModelInfo = {
        model = "art/pickups/health/megaHealth_head_a.lwo";
        contributesToLightProbeGen = false;
        noAmbient = true;
        scale = {
          x = 1;
          y = 1;
          z = 1;
        }
        emissiveColor = {
          r = 0;
```

```

    g = 0.913725019;
}
emissiveScale = 3;
}
spawn_statIncreases = {
    num = 1;
    item[0] = {
        stat = "STAT_ITEMS_SPAWNED";
        increase = 1;
    }
}
equipOnPickup = true;
lootStyle = "LOOT_TOUCH";
triggerDef = "trigger/props/megahealth";
isStatic = true;
canBePossessed = true;
fxDecl = "pickups/megahealth_head";
useableComponentDecl = "mega_health";
sound_spawn = "play_pickup_loop_05";
sound_stop = "stop_pickup_loop_05";
pickup_statIncreases = {
    num = 1;
    item[0] = {
        stat = "STAT_HEALTH_PICKUP";
        increase = 1;
    }
}
spawnPosition = { // Be sure to set the spawn position
    x = -11.11;
    y = -11.11;
    z = -11.11;
}
spawnOrientation = { // Be sure to set the spawn orientation
    mat = {
        mat[0] = {
            x = -1;
            y = 0;
            z = 0;
        }
    }
}

```

```

mat[1] = {
  x = 0;
  y = -1;
  z = 0;
}
}
}

targets = {
  num = 1;
  item[0] = "deathmatch_pickup_mega_health_1_respawn_relay"; // Activates relay to start
respawn process
}
isLootDrop = true; // Must be true
}
}

```

idMover

Optional, but will fix the entity offset for the pickup. However, the pickup's spawn orientation will revert to its default value.

```

entity {
  entityDef deathmatch_pickup_mega_health_1_offset_mover {
    inherit = "func/mover";
    class = "idMover";
    expandInheritance = false;
    poolCount = 0;
    poolGranularity = 2;
    networkReplicated = true;
    disableAIPooling = false;
    edit = {
      spawnPosition = { // Be sure to set the spawn position
        x = -11.11;
        y = -11.11;
        z = -11.11;
      }
      spawnOrientation = { // The spawn orientation does not matter

```

```

mat = {
  mat[0] = {
    x = -1;
    y = 0;
    z = 0;
  }
  mat[1] = {
    x = 0;
    y = -1;
    z = 0;
  }
}

renderModelInfo = {
  model = NULL;
}

clipModelInfo = {
  clipModelName = NULL;
}
}

```

Respawning Pickups

Target Spawn

```

entity {
  entityDef deathmatch_pickup_mega_health_1_respawn {
    inherit = "target/spawn";
    class = "idTarget_Spawn";
    expandInheritance = false;
    poolCount = 0;
    poolGranularity = 2;
    networkReplicated = false; // Must be false
    disableAIPooling = false;
    edit = {

```

```

flags = {
    noFlood = true;
}

spawnConditions = {
    maxCount = 0;
    reuseDelaySec = 0;
    doBoundsTest = false;
    boundsTestType = "BOUNDSTEST_NONE";
    fovCheck = 0;
    minDistance = 0;
    maxDistance = 0;
    neighborSpawnerDistance = -1;
    LOS_Test = "LOS_NONE";
    playerToTest = "PLAYER_SP";
    conditionProxy = "";
}

spawnEditableShared = {
    groupName = "";
    deathTrigger = "";
    coverRadius = 0;
    maxEnemyCoverDistance = 0;
}

entityDefs = {
    num = 1;
    item[ 0 ] = {
        name = "deathmatch_pickup_mega_health_1"; // Activates respawning pickup
    }
}

conductorEntityAIType = "SPAWN_AI_TYPE_ANY";

initialEntityDefs = {
    num = 0;
}

spawnEditable = {
    spawnAt = "";
    copyTargets = false;
    additionalTargets = {
        num = 0;
    }
}

overwriteTraversalFlags = true;

```

```

    traversalClassFlags = "CLASS_A";

    combatHintClass = "CLASS_ALL";

    spawnAnim = "";

    aiStateOverride = "AI_OVERRIDE_TELEPORT";

    initialTargetOverride = "";

}

portal = "";

targetSpawnParent = "";

disablePooling = false;

spawnPosition = { // Be sure to set the spawn position
    x = -11.11;
    y = -11.11;
    z = -11.11;
}

spawnOrientation = { // Be sure to set the spawn orientation
    mat = {
        mat[0] = {
            x = -1;
            y = 0;
            z = 0;
        }
        mat[1] = {
            x = 0;
            y = -1;
            z = 0;
        }
    }
}
}

```

```
entity {
  entityDef deathmatch_pickup_mega_health_1_respawn_relay {
    inherit = "target/relay";
    class = "idTarget_Count";
    expandInheritance = false;
  }
}
```

```
poolCount = 0;
poolGranularity = 2;
networkReplicated = false; // Must be false
disableAIPooling = false;
edit = {
  flags = {
    noFlood = true;
  }
  networkSerializeTransforms = false;
  count = 1; // Activates on one count
  spawnPosition = { // Spawn position is not important
    x = 1;
    y = 1;
    z = 1;
  }
  targets = {
    num = 1;
    item[0] = "deathmatch_pickup_mega_health_1_respawn"; // Activates target spawn
  }
  delay = 15; // Must have a delay
  repeat = true; // Can activate again
}
}
```

Revision #12

Created 22 August 2023 20:09:46 by Velser

Updated 12 February 2024 20:38:07 by Velser