Contents


```
%Manages the particles representing the rocket's exhaust as they interact
%with the ground.
classdef ExhaustMgr < handle
    properties
        particleList %Array of particles in use
        rocket %rocket to referene
        particleSpawnTimer %Counts down. When zero, spawn a new particle.
    end

    methods
```

Constructor

```
        %Creates an ExhaustMgr object and initializes its particle list
        function obj = ExhaustMgr(rocket)
            %Creates an empty 0x0 array of SpriteKit.Sprites that will be
            %filled with the particles.
            obj.particleList = SpriteKit.Sprite.empty();
            obj.rocket = rocket;
            obj.particleSpawnTimer = Const.particleSpawnTime;

            %disp(obj.particleList);
            %Initialize the particle list
            for i = 1:Const.numParticles
                %part1, part2, etc.
                particleName = sprintf('part_%d', i);

                ithParticle = SpriteKit.Sprite(particleName);

                ithParticle.Scale = Const.particleScale;

                initState(ithParticle, 'on', Const.exhaustImg, true);
                initState(ithParticle, 'off', Const.noneImg, true);
                initState(ithParticle, 'dirt', Const.dirtImg, true);

                addprop(ithParticle, 'velocity');
                addprop(ithParticle, 'age');

                ithParticle.velocity = [0,0]; %Initialize the vector
                ithParticle.Location = Const.defaultParticlePos;
                ithParticle.Depth = 1; %Depth 1 for default, it gets randomized later.
                ithParticle.age = Const.particleDespawnedAge;
                obj.particleList(i) = ithParticle;
            end

        end
```

```
Not enough input arguments.

Error in ExhaustMgr (line 17)
            obj.rocket = rocket;
```

killAllParticles

```
        %disables all managed particles.
        function killAllParticles(obj)
            %Iterate through the list of particles and disable all.
            for currentParticle = obj.particleList
                despawnParticle(obj, currentParticle);
            end
        end
```

spawnParticle

```
        %spawns a particle if possible.
        %position is the position to be spawned at, as [x,y]
        %velocity is [x,y] velocity in pixels per frame to be spawned with.
        function spawnParticle(obj, position, velocity)
            %Find the index of the oldest particle.
            targetIndex = 1;
            oldestAge = 0;
            for i = 1:length(obj.particleList)
                currentParticle = obj.particleList(i);
                if currentParticle.age > oldestAge
                    targetIndex = i;
                    oldestAge = currentParticle.age;
                end
            end
            targetParticle = obj.particleList(targetIndex);

            targetParticle.State = 'on'; %Enable the target particle
            targetParticle.age = 0;
            targetParticle.Location = position;
            targetParticle.velocity = velocity;
            targetParticle.Angle = randi(359);
            targetParticle.Scale = Const.particleScale;

            %Randomized depth doesn't actually look much better, and it
            %means more matrix math, so it impacts performance.
            %{
            %Randomize the depth of the particle to create a 3D-ish
            %illusion. The depth is not ever equal to 5, as this is the
            %depth of the rocket and cow.
            if rand <= 0.5
                targetParticle.Depth = randi(4);
            else
                targetParticle.Depth = randi(4) + Const.foregroundDepth;
            end

            %}

            %Reset the particle spawn timer.
            obj.particleSpawnTimer = Const.particleSpawnTime;
        end
```

despawnParticle

```
%despawns the given particle.
%Note that particle should be a
%particle Sprite object, not a particle index.
function despawnParticle(~, particle)
    particle.State = 'off';
    %Setting the age to a very high number allows this to be one of
    %the first particles when sorted by age.
    particle.age = Const.particleDespawnedAge;
    particle.Location = [0,0];
end
```

spawnParticleFromEngine

```
%spawns a new particle from the rocket's engine
function spawnParticleFromEngine(obj)
    if obj.rocket.throttle >= Const.throttle0cutoff
        %These are easier to refer to
        rocketLoc = obj.rocket.Location;
        rocketAngle = obj.rocket.Angle;

        %This offsets the particle's spawn so that it is emerging from
        %the engine and not the center of the rocket.

        %Horizontal distance
        engineOffsetX = -1 * Const.particleOffsetDistance * sind(rocketAngle);
        %Vertical distance
        engineOffsetY = Const.particleOffsetDistance * cosd(rocketAngle);

        %Location to spawn the particle
        spawnPos = rocketLoc + [engineOffsetX, engineOffsetY];

        %Add a random positive offset to the direction of the velocity
        %to make the exhaust look less linear
        angleOffset = randi(Const.particleAngleSpread);

        %Randomize the direction of the offset (clockwise or
        %counterclockwise)
        %velAngle is the direction of the particle's velocity vector.
        if rand <= 0.5
            velDir = rocketAngle - angleOffset;
        else
            velDir = rocketAngle + angleOffset;
        end

        %The magnitude of the velocity for the particle to be spawned with
        %This is based on the rocket's throttle setting.
        spawnSpeed = Const.particleMaxVelocity * obj.rocket.throttle;

        %Multiply the spawn speed by the unit vector in the spawn
        %direction.
        spawnVelocity = spawnSpeed * [sind(velDir), -cosd(velDir)];

        spawnParticle(obj, spawnPos, spawnVelocity);
    end
end
```

updateParticles

```
        %Scrolls all particles assigned to the manager and updates them.
        %Takes [x,y] particle scroll distance.
        %Spawns new particles if appropriate.
        function updateParticles(obj, particleScrollDist)
            %Scroll particles
            %Move every particle in this manager's list by the distance
            %specified
            for currentParticle = obj.particleList
                %Only update these if the particle is actually enabled.
                if currentParticle.State ~= "off"
                %Move each particle according to its velocity.
                newLocation = currentParticle.Location...
                        + (currentParticle.velocity * Const.pixelsPerMeter...
                        * Const.frameTime);

                %Scroll the particle as needed.
                newLocation = newLocation...
                    + particleScrollDist * Const.pixelsPerMeter;

                %Check if the particle has hit the ground.
                if newLocation(2) <= Const.zeroAlt
                    %Force the particle above ground.
                    newLocation(2) = Const.zeroAlt;

                    %Make its velocity in the vertical direction positive
                    currentParticle.velocity(2) = abs(currentParticle.velocity(2));

                    %Slow down the particle some to make its behavior more
                    %interesting
                    currentParticle.velocity = currentParticle.velocity...
                        .* Const.particleGroundDrag;
                    if currentParticle.State ~= "dirt"
                        currentParticle.State = 'dirt'; %Change to dirt particle
                    end
                end

                currentParticle.Location = newLocation;

                %Reduce the particles' speed slightly to simulate drag.
                currentParticle.velocity = currentParticle.velocity * Const.particleAirDrag;
                %Apply gravity to the particles.
                currentParticle.velocity = currentParticle.velocity...
                    + Const.particleGravity * Const.frameTime;

                currentParticle.Scale = currentParticle.Scale + Const.particleScaleInc;

                %Despawn checks
                %If the particle is scrolled off screen, disable it.
                if (currentParticle.Location(1) <= 0 ||... %left
                    currentParticle.Location(1) >= Const.windowSize(1) ||... %right
                    currentParticle.Location(2) <= 0 ||... %bottom
                    currentParticle.Location(2) >= Const.windowSize(2)) %top

                    despawnParticle(obj, currentParticle); %disable particle
                end

                %Increment the age
                currentParticle.age = currentParticle.age + 1;
```

```
                %If the current particle's age is greater than the maximum,
                %disable it.
                if currentParticle.age >= Const.particleMaxAge
                    despawnParticle(obj, currentParticle); %disable particle
                end
                end
            end

            if obj.rocket.gameState == "play" && obj.rocket.propMass > 0
                %Spawn new particles if needed
                obj.particleSpawnTimer = obj.particleSpawnTimer - 1;
                if obj.particleSpawnTimer <= 0
                    %try to spawn a particle from the engine
                    spawnParticleFromEngine(obj);
                end
            end
        end
```

```
    end
end
```

Published with MATLAB® R2021b