Game Objective:

The objective is to collect/rescue the three missing baby slowpokes and return them to their mom, who is frozen and needs them to be revived. The user must avoid a hot air balloon that is following the user-controlled helicopter or else lose.

Note: Many trees are placed randomly so if any obstructs your vision please refresh the window

---

Physics with collisions - Adopt your 2D physics so that they also meaningfully work in the 3D environment. Specifically, be able to navigate with your avatar. Provide some game mechanism that requires handling collisions between spherical colliders.

The user navigates the helicopter.

Keys:

J = left

L = Right

K = Forward

I = Back

U = Down

O = Up

When the helicopter collides with a slowpoke, the slowpoke disappears/is rescued.

Illumination - A single directional light source and a few lesser point lights should illuminate your game world. Objects should be shaded with diffuse and Phong-Blinn shading, possibly combined with texturing.

There is a single directional light source. There are two point lights on the frozen slowpoke mom. One point light on the slowpoke is moving, which makes for a cool visual effect. Objects shaded with both diffuse and Phong-Blinn shading. Some objects like the helicopter and tree are shinier than others.

Sky -Display an environment map as a background to all rendered objects. The visible part of the sky should properly depend on camera orientation.

Barren landscape background texture

Environment mapping -Have at least one object reflect the environment (but not other objects).

The hot air balloon attached to a slowpoke is reflecting the environment. The other balloons are also, but with procedural normal mapping too

AI (requires Physics) -Some type of objects should move towards the avatar. Use acceleration and drag. What (if anything) happens on impact is up to you. Some different objects should visit predefined checkpoints in order. Use acceleration and drag. A checkpoint should be considered visited when the object comes within some proximity radius. Bonus points if the moving object has an identifiable 'ahead' direction, and when moving towards the avatar or a checkpoint, it moves ahead.

A hot air balloon follows the helicopter, which the helicopter must avoid.

The car in the background moves towards the position of the tree to its left, hits it, and the tree falls.

When the helicopter comes close to the slowpoke on the ground, the slowpoke heads toward a different position.

Rotating parts -The avatar, or some other prominent object, should have at least one rotating part (propeller, wheel, etc.).

The rotor on top of the helicopter rotates. It slows down when it hits the ground(zx plane) and speeds up when it lifts off

Frenet frame (assumes airborne avatar) -Position is physically animated, orientation is not. In case of a fixed-winged aircraft: nose points the same direction as the velocity vector, wing points into direction velocity-cross-acceleration-normalized, and tail fin points into direction nose-cross-wing-normalized.

Could not figure out

Helicam -The camera should always look at the avatar, or at the point where the avatar is immediately heading (position + velocity * something). The distance from the avatar needs to be kept reasonably constant.

The camera looks towards the helicopter and is a little ahead of it.

Plane-projected shadows -Shadows of objects should appear on the ground to give a sense of altitude (or the lack of it). Draw objects in black, flattened to the ground along the light direction, slightly above the ground plane.

The trees and slowpokes have shadows, all affected by the same light direction.

Procedural texturing -Have at least one object with procedural solid texturing (e.g. wood or marble)---i.e. the pixel shader should compute color from world space position using some formula.

The trees have procedural texturing in the form of stripes.

Procedural normal mapping - At least one object should have a bumpy surface achieved by its normals being perturbed using procedural solid texturing---i.e. a smooth noise value that depends on the world space position should be added to the world space normal otherwise computed. It is recommended that this object is environment mapped, for best effect.

The two hot air balloons have bumpy surfaces. The surface of the moving balloon changes as the noise function depends on world position.