Essential Classes for a 3D Game in JavaScript (Plain Text for RAG)

1. GameManager (Game Engine)
- Core class that initializes and updates the game loop.
- Handles input, timing, and updating all game objects.

Example:
```
class GameManager {
  constructor() {
    this.scene = new THREE.Scene();
    this.clock = new THREE.Clock();
    this.objects = [];
  }

  update() {
    const delta = this.clock.getDelta();
    this.objects.forEach(obj => obj.update(delta));
  }
}
```

2. Player (Character)
- Represents the player's entity.
- Manages movement, animations, controls, and health.

Example:
```
class Player {
  constructor(model) {
    this.model = model;
    this.velocity = new THREE.Vector3();
  }

  update(delta) {
    // Apply movement and controls
  }
}
```

3. InputManager
- Handles keyboard, mouse, and gamepad input.
- Decouples raw input from game logic.

Example:
```
class InputManager {
  constructor() {
    this.keys = {};
    window.addEventListener('keydown', e => this.keys[e.code] = true);
    window.addEventListener('keyup', e => this.keys[e.code] = false);
  }

  isKeyPressed(key) {
    return this.keys[key];
  }
}
```

4. GameObject (Entity)
- Base class for any object in the game (enemies, items, props).
- Includes position, collision, and behavior logic.

Example:

```
class GameObject {
  constructor(model) {
    this.model = model;
    this.position = model.position;
  }

  update(delta) {
    // Default behavior
  }
}
```

5. CollisionManager (Physics)
- Manages object collisions, gravity, and movement constraints.

Example:

```
class CollisionManager {
  checkCollision(objA, objB) {
    const boxA = new THREE.Box3().setFromObject(objA.model);
    const boxB = new THREE.Box3().setFromObject(objB.model);
    return boxA.intersectsBox(boxB);
  }
}
```