Dead or Alive!

1. Touch and be alive!

1.1. Introduction

You are the son of Death and your father just handed you a planet that he destroyed just for you to play with. He wants you to train your powers by bringing objects to life with souls that still remain in the world, and then killing them with your death aura.

Dead or Alive! is a 3D beat-em-up style game with a twist: Do you want them dead or alive?

1.2. Vision Statement

The player should be delivered the following experiences:

- **Effortless** play originating from a simple control scheme.
- **Unique** experience – the player has to bring the objects to life before beating them.
- **Exploration** of different environments.
- **Choice** of life and death.

1.3. Rough Plot

You start the game (to be developed)

2. Core Gameplay

2.1. Core Mechanics

The core mechanics in DOA consists of three elements:

- **Touch of life** that makes all the objects the player touches come to life.
- **Aura of Death** that kills the objects that have come to life when they come in contact with the player again.
- **Character** development of your avatar as you kill or save lives.

2.2. Avatar

2.2.1 Overview

The player's Avatar is the son of Death with the following abilities:

- **Move** around the environment. The avatar moves according to the directional buttons pressed by the player.
- **Touch** the objects in the environment. When the avatar is near to an object, she can touch the object and bring it to life.
- **Aura of Death** that is always present on the avatar. Objects that come in contact with the player will die.

2.3. Controls

Directional Keys

The avatar moves in the corresponding direction of the directional keys that are pressed.

Touch

The avatar makes a touch action and any objects that he contacts are brought to life.

Aura of Death

The avatar changes mode and releases his aura of death. In this mode, the avatar kills everything he touches.

3. Systematic breakdown of components

3.1. 3D Renderer with Pinocchio integrated

This allows the artists to input their art assets into the framework of the game system with the auto-rigging system "Pinocchio" integrated to the assets.

3.2. Level editor

This allows the game designer to design levels make changes to it easily.

3.3. Collision system

A collision system is required for interaction between the player, objects, and obstacles in the game environment.

3.4. Pathfinding algorithms

Objects that have come to life have different object behaviours that will require various pathfinding algorithms.

3.5. Object state machine

The system must be able to record what state the object is in so as to keep a scoring system.

3.6. Scoring system

Based on the state of the objects, this system will keep a score of the amount of objects you have brought to life and the amount of objects you have killed in total.

3.7. Sound library integration system

This system allows the easy implementation of sound library that the audio engineer has created into the game system ready for use within the game.

4. Suggested Game Flow Diagram