BRASH

by Ethan Ha and Steven Ho

CSC 165-02

Screenshots

Instructions

Server Side:

- Compile the .java files by running compileServer.bat
- Run the server by starting runServer.bat inside project root folder.

Client Side:

- Make sure runClient.bat has the correct IP address of the hosted computer and that it is in the same network for any clients
- Compile the .java files by running compileClient.bat
- And then lastly, start your game with runClient.bat

Special Device Requirements

- No special device requirements

Gameplay

Brash is a game that is related to other similar games that has the game style of breaking an object and running away from enemy guarding such objects. As from the title of the game "Brash", it involves players destroying as many box objects as possible all the while avoiding this monstrous duck that safeguards these boxes. Getting caught by this duck results in the player losing and being kicked out of the game. Other players on the server could either help or hinder other players as they can help distract the duck or lure the duck towards you. The number of boxes broken is how players earn points and if all the boxes are broken, the player with the highest score wins.

Controls

Keyboard:

- Movement:
    - W – Move avatar forward
    - S – Move avatar backward
    - A – Turn avatar to the left
    - D – Turn avatar to the right
- Camera Movement:
    - UP ARROW – Elevate the camera upwards
    - DOWN ARROW – Elevates the camera downwards
    - LEFT ARROW – Rotates the camera to the left
    - RIGHT ARROW – Rotates the camera to the right
    - PAGE UP – Zooms out the player's camera
    - PAGE DOWN – Zoom in the player's camera
- Actions:
    - SPACE – Makes the avatar jump
    - F – Avatar punch

    - ESC – Close game client

Gamepad:

- Movement:
    - Z-AXIS Negative – Moves avatar forward
    - Z-AXIS Positive – Moves avatar backward
- Camera Movement:
    - Left Stick
        - X-AXIS Negative – Elevate camera up
        - X-AXIS Positive – Elevate camera down
    - Right Stick
        - X-ROTATION Negative – Rotate camera to the left
        - X-ROTATION Positive – Rotate camera to the right
        - Y-ROTATION Negative – Zooms in camera
        - Y-ROTATION Positive – Zooms out camera
- Actions:
    - BUTTON 1 – Makes avatar jump
    - BUTTON 3 – Avatar punch
    - BUTTON 8 – Close game client

Scripting

- Initializing
    - Player position
    - Ghost position
    - NPC position
    - Jukebox position
- Function
    - Increase player score

Changes made to Network Protocol

- Client receives messages from server that:
    - Tells client to make a certain number of boxes with the servers specified amount and random box location
    - Receives information if a box has been destroyed
    - Tells the client that NPC is made with it moving back and forth at a specified location
    - Tells the client of other ghost avatar animation, location, rotation
    - Updating the HUD to include the ghost avatar points
- Server receives messages from client:
    - Receiving a message that a new client wants to join
    - Receiving bye message of the client leaving the server

- Receiving any client's movement, animation, rotation
- Receiving packet from client if server needs to send a create new player score to every client
- Getting packets that askes the server if a client needs an NPC, boxes
- Handling incoming packets if the player/ghost is near the NPC

Changes/Additions made in TAGE

- Added ProtocolClient.java to the Networking/Client folder

Statement

- 1) Genre: Casual, Adventure Game
- 2) Theme: Fantasy
- 3) Dimensionality: 3D
- 4) Activities: Exploration, Combat

Project Requirements Satisfaction

- External models:
  - Different models are viewable on screen as when 2 players join, they each have their own model skin. Additionally, the NPC that is guarding the boxes has its own model and texture in game.
- Networked Multiplayer:
  - Two or more players are able to join the same world. Each player must be connected to one IP address and the other player will appear as a ghost avatar. If you want to just play solo, the player can do so by not sharing out their IP address.
- Scripting:
  - In our game, scripting acts as a way to set the player/ghost location from the start and to also update the player's point counters when a player earns a point.
- Skybox and Terrain:
  - Our game has a custom skybox that can be seen all around the player. Moreover, when loading into the game, the player will be on the world plane. All around the player will be the terrain created by the gray-scaling in which the player can free-roam and explore.
- Lights:
  - Spotlight on jukebox object that plays the background music.
  - Global light
- HUD:
  - HUD in our game displays the amount of time the game has been running and the points of the players.
- 3D Sound:
  - Upon starting the game, the background music will be playing, it is also positional. Walking, punching, or jumping will have subtle sound queues when performing such actions.

- Hierarchical SceneGraph:
    - The only hierarchical relationship we have in our game is a hierarchical object with the player's avatar and the crown. The crown is a child object of the avatar and will rotate and move accordingly to the avatar.
- Animation:
    - Animation is viewable when walking, punching and jumping in game.
- NPCs:
    - We have one NPC/AI in our game and it is the duck that defends the boxes. This AI will walk back and forth, but will walk towards the player's direction once in close proximity of the player.
- Physics:
    - Our physics in this game is the player's ability to jump, as seen from how the player launches up and how they fall down to the ground because of gravity.

Requirements not implemented/satisfied

- Avatar Selection
- Animation for Duck NPC/AI
- Proper win/lose screen
- Gamepad controls not working
- Ghost animation now working properly
- Collision detection for boxes and avatar

Beyond the Requirements

- No features beyond the requirements

Contributions

Ethan Ha

- Models – Player, Ghost, NPC
- Terrain
- Sound (BGM, sound effects)
- Animation
- Skybox
- Random box spawn

Steven Ho

- Model – Crown
- Scripting
- Networking
- Ghost avatar animating
- Physics
- NPCs/AI

Evidence to Use 3rd Party things in our Game

- Models:
  - crown.obj – Steven's creation in Blender
  - duck.obj – Ethan's creation in Blender
  - player.obj – Ethan's creation in Blender
- Animations:
  - Duck files – Ethan's creation in Blender
  - Player files – Ethan's creation in Blender
- Textures:
  - crate.png – Ethan's creation in Paint
  - crown_texture.png – Steven's creation using Blender UV export and Paint
  - duck_uv.png – Ethan's creation using Blender UV export and Paint
  - ghost_uv.png – Ethan's creation using Blender UV export and Paint
  - grass.png – Ethan's creation in Paint
  - hill.png – Ethan's creation in Paint
  - jukebox.png – Ethan's creation in Paint
  - player_uv.png - Ethan's creation using Blender UV export and Paint
- Skybox Cube Map:
  - sereneClouds: xn.jpg, xp.jpg, yn.jpg, yp.jpg, zn.jpg, zp.jpg
  - All images excerpted from AI Skybox generation at Blockade Labs (Link), according to Stable Diffusion, CompVis allows free use of these generated image outputs.

Licenses

By using Skybox AI, you give Blockade Games, Inc. and 330AI Innovations, Inc. the ongoing permission to use any images you create with the tool for any purpose, including commercial and non-commercial use. This permission is perpetual, which means it lasts indefinitely, and non-exclusive, which means you still own the images and can use them yourself. This permission also allows us to improve our models and algorithms using the images you create, and to use the images for promotional materials, such as in social media or marketing campaigns.

Skybox AI uses a modified version of Stable Diffusion. Stable Diffusion is open access and available to all, with a CreativeML OpenRAIL-M license further specifying rights and usage.

The CreativeML OpenRAIL License specifies:

You can't use the model to deliberately produce nor share illegal or harmful outputs or content.

CompVis claims no rights on the outputs you generate, you are free to use them and are accountable for their use which must not go against the provisions set in the license. You may re-distribute the weights and use the model commercially and/or as a service. If you do, please be aware you have to include the same use restrictions as the ones in the license and share a copy of the CreativeML OpenRAIL-M to all your users (please read the license entirely and carefully).

Please read the full license here:
https://huggingface.co/spaces/CompVis/stable-diffusion-license

RVR-5029 Lab Machines

- TEKKEN
- SPACEQUEST