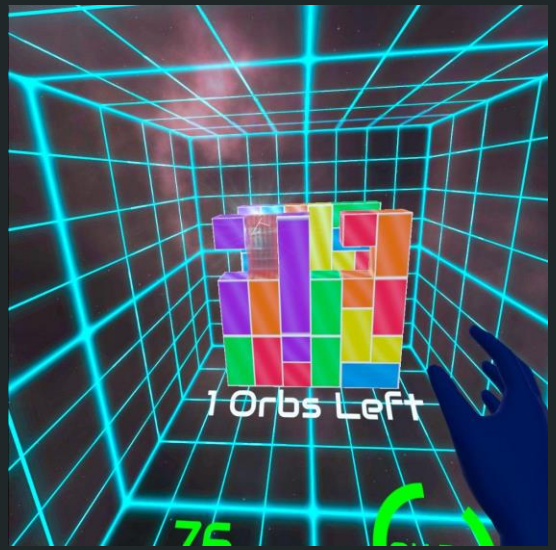
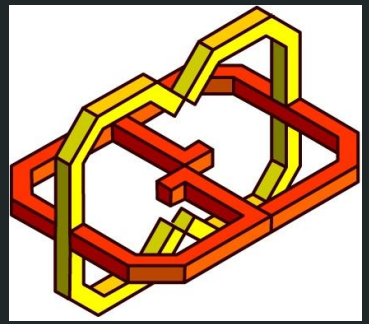
Brick Smash VR

Students: Daniel Cohen, Meir Friedmann

Supervisors: Boaz Sternfeld, Yaron Honen



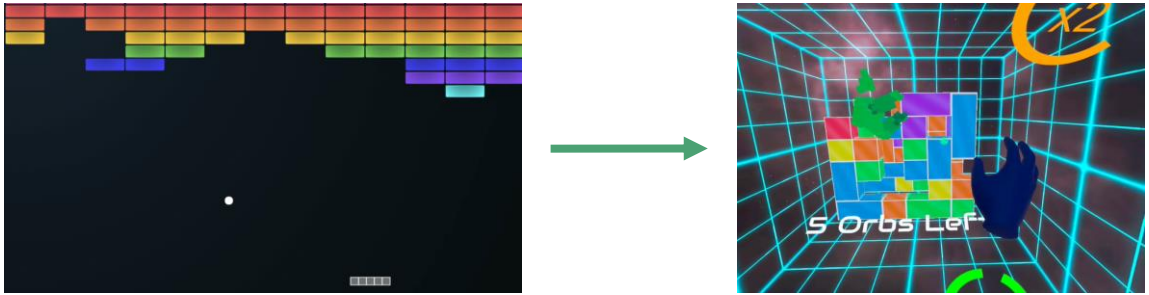
Completed in collaboration with the Center for Graphics and Geometric Computing as well as with the Geometric Image Processing Lab at the Technion – Israel Institute of Technology



Introduction

Brick Smash VR is a VR game inspired by the widely popular classic video games *Breakout* and *Arkanoid,* in which the player tries to break all the bricks by using a paddle to deflect a ball at them. Our goal was to take these classics and transform them into an immersive VR experience, but with a few key enhancements:

- The player plays an active physical role in the game as the paddle
- When hit with a ball, the bricks actually fracture and explode into fragments instead of just disappearing
- The fragments of destroyed bricks play an important role in the game
- The walls of bricks move



Features

- Custom brick fracturing dependent on impact force and position
- Brick walls which can move vertically and horizontally, as well as being rotated
- Five power-ups
- Sound effects, music, and visual effects
- Persistent high score system
- Three difficulty levels (beginner, normal, expert)
- Racket and ball spawn locations swappable (left/right hand use)
- Adjustable music and SFX volume
- Game can be paused

Hardware

The game was implemented as an android application for the Meta Quest 2 (with both controllers).

Technologies and Platforms Used

- Unity 2021.3.8
- Unity Open XR Plugin
  - OpenXR is an open, royalty-free standard developed by Khronos that aims to simplify AR/VR development by allowing developers to seamlessly target a wide range of AR/VR devices.
- Unity XR interaction Toolkit
  - A high-level, component-based, interaction system for creating VR and AR experiences. It provides a framework that makes 3D and UI interactions available from Unity input events.
- Visual Studio Code
- Adobe Photoshop

Gameplay

- The player starts with a certain number of orbs (balls)
- The objective of the game is to break all the bricks before the time runs out
- Along the way, the player can earn points by collecting the fragments from broken bricks
- The larger the fragment, the more points collected
- The game is over if the player runs out of balls or the time runs out before all the bricks are broken
- There are five power-ups that can be randomly generated and can be collected by the player (not all of them beneficial to the player):
  - Double points
  - Negative points
  - Extra ball
  - Move walls vertically / horizontally
  - Rotate walls
- Two methods for controlling the balls in the arena
  - Catching and throwing with hands
  - Hitting with a racket
- Player has two axis of movement (up/down and left/right)

Wall Design

- Two walls
- Brick colors chosen randomly from set of HDR colors
- Algorithm used to tile each wall with randomly sized bricks

Brick Fracture Process

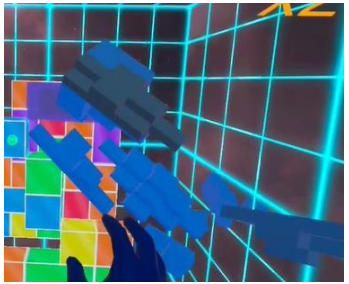
Ball Hits Brick → Impact Crater Created → Fracture Lines Created → Fragments Created → Brick Explosion

Voxels are removed at position of ball impact to create a crater. The size of the crater is dependent on the force of the impact.

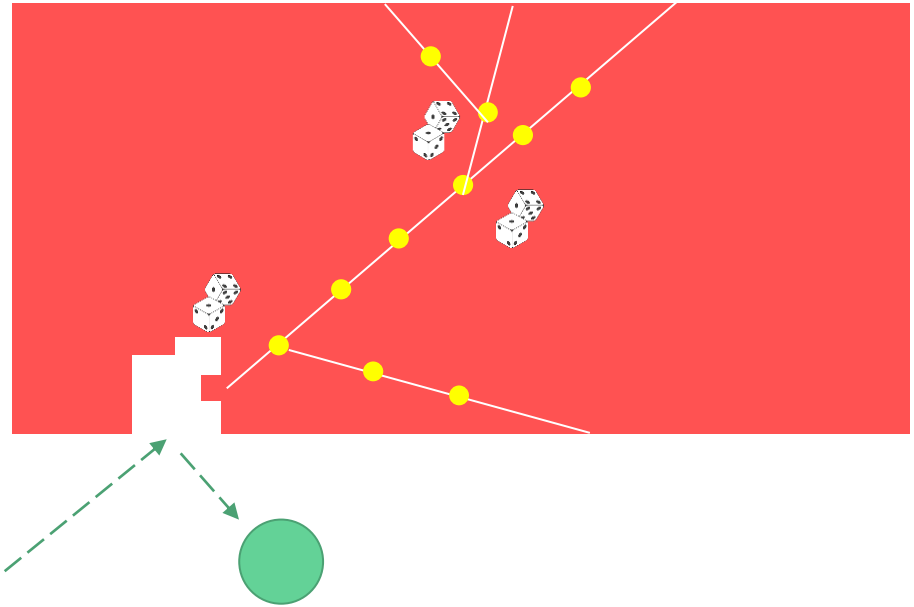
A fracture line is created originating at the collision point and extending in the same direction as the velocity vector of the ball at the point of contact. Along this fracture line at uniform intervals nodes are generated, from which additional fracture lines are forked with random probability and with a randomly generated new direction.

Voxels are grouped into fragments based on the voxel's distance to the impact point as well as on the impact force.

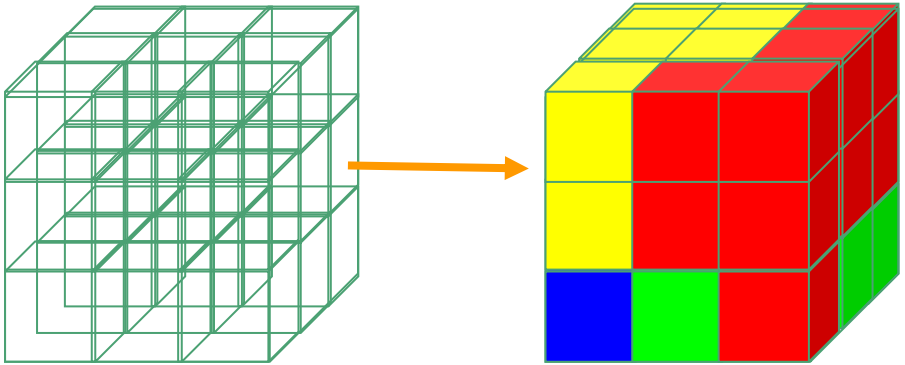
Fragments have force applied to them based on their position relative to the closest fracture line as-well as the force of impact.



Brick Fracture Lines 2D Visualization



Visualization of Brick Fragment Formation



General Challenges

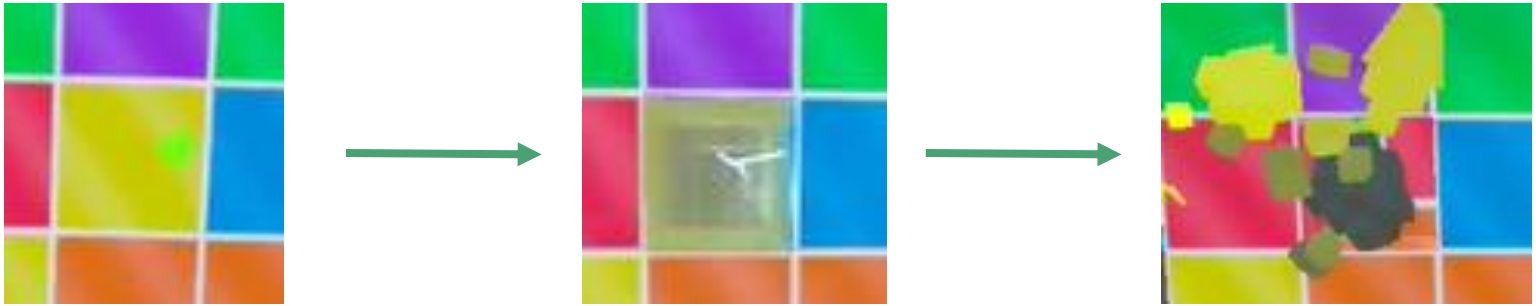
- Learning Unity platform and more specifically Unity VR frameworks from scratch
- Opening scene caused motion sickness and was not realistic
  - Decided to skip adding opening scene
- Ball accelerated to uncontrollable speeds
  - Added speed limiter to ball

Overcoming Processing Limitations

- Making use of HDR color instead of post-processing layers
- Object pre-initialization and pooling
- Limiting voxel count per brick
- Prohibiting simultaneous brick collisions

Realism Versus Visual Appeal

- Brick fragments were getting stuck and not exploding
  - Disabled physics interactions between fragments and surrounding bricks
- Brick fracture lines were very hard to see
  - Make brick temporarily transparent when drawing fracture lines



Challenges During Brick Fragmentation

- Adjust Parameters to optimize performance / visuals ratio:
  - Fracture line fork probability
  - Voxel count
  - Fragment size min/max limits
  - Fracture line depth
- Dealing with case of ball hitting edge of brick
  - Manipulate angle of ball velocity

Conclusion

Beyond supplying us with more experience with programming in C# as well as with working with the Unity platform, this project has opened our eyes to the depth and complexity involved with game design in general and specifically in VR, and has motivated us to continuing exploring this very interesting field!