Report
Computer Graphic Group 8
Simulate Horse Racing Chess

1. Team member

	Full Name	Student ID	E-mail
1	Đồng Xuân Thủy (Leader)	10020347	thuydx_55@vnu.edu.vn
2	Chu Việt Anh	10020002	anhcv_550@vnu.edu.vn
3	Đặng Trần Thái	10020325	thaidt_55@vnu.edu.vn
4	Nguyễn Thế Tùng	10020417	tungnt_55@vnu.edu.vn
5	Trịnh Văn Tú (Designer)	10020427	tunv_55@vnu.edu.vn

2. Describe Project

Simulate a board for 2-4 players
Simulate moving pieces and attack pieces
Change the viewport arbitrary

3. Project github source

https://github.com/thuydx55/CoCaNgua

4. Project Plan

From 11/3

	Week	Content	Work	Member
1	1	Prepare	Plan, find idea, assign work	All members
2	2	Display Model in 3D view Display temporary models and textures	Load Model from *.obj file Load texture Create temporary models in 3ds max Draw model and texture in 3D view	Đồng Xuân Thủy
3	3	Classify object Library for 3D objects	Class Model, Piece, Die, Board, Rock Function get/set position, get/set Anchor Point, …	Đồng Xuân Thủy
4	4	Light & Camera, matlib Mouse clicking Move Piece Player turn Game Logic	Create class Light and Camera Add matlib library Mouse click to objects Piece move and jump Next turn if current player can't move Specify type of move: start, attack, start and attack, move home from road, move inside home, can't move	Đồng Xuân Thủy Nguyễn Thế Tùng Chu Việt Anh
5	5	Die Model Rotate Die Game Logic (continue) Input Manager	Final Die Model Throw Die (rotate die randomly) and get the number Game Logic according rules Manage Key press, mouse click in	Đồng Xuân Thủy Chu Việt Anh Trịnh Văn Tú

			specify scene	
6	6	Shadow & Highlight Library for 2D scenes Game Logic (continue)	Create shadow for object Class Sprite2D, Button, ToggleButton, RadioGroup ProgressTimerSprite2D Identify winner	Đặng Trần Thái Đồng Xuân Thủy Chu Việt Anh Trịnh Văn Tú
7	7	Specify Scenes Integrate UI Visual effects Camera (continue) Sound	Scenes: Loading, Main Menu, Option, About, Select Players, Game Add final textures Visual effect: slow time, direction of pieces… Camera auto rotate Integrate sound	Đồng Xuân Thủy Trịnh Văn Tú Đặng Trần Thái Nguyễn Thế Tùng
8	8	Test and Debug Comments	Testing Debugging Comments Final Optimization.	All members

I. Part of game has been completed

1. Model Class

- Load data from obj file and store it in memory.
    - Vector of vertices.
    - Vector of meshes – each mesh is a triangle; contains indexes of 3 vertices in vector vertices.
    - Vector of materials – each mesh contains ambient light, diffuse light, specular light, shininess, alpha, name of texture.
- Draw model with texture.
- Change color of material by changing ambient and diffuse light.

2. Camera

- Using Spherical Coordinates.
- Zoom, Rotate Camera.

3. Game Scene

- Initialize all model used in game.
- Draw scene and die.
- After die is thrown, we get the number appear on top face of die. Each Piece will predict what type of moving is and which field it's will be in. Highlight all Pieces can move.
- Identify which Pieces is clicked, then move it to predict position.
- Control die appearing.
- Players turn.

4. Die Class

- Die also use spherical coordinates. ϕ (phi) & θ (theta).

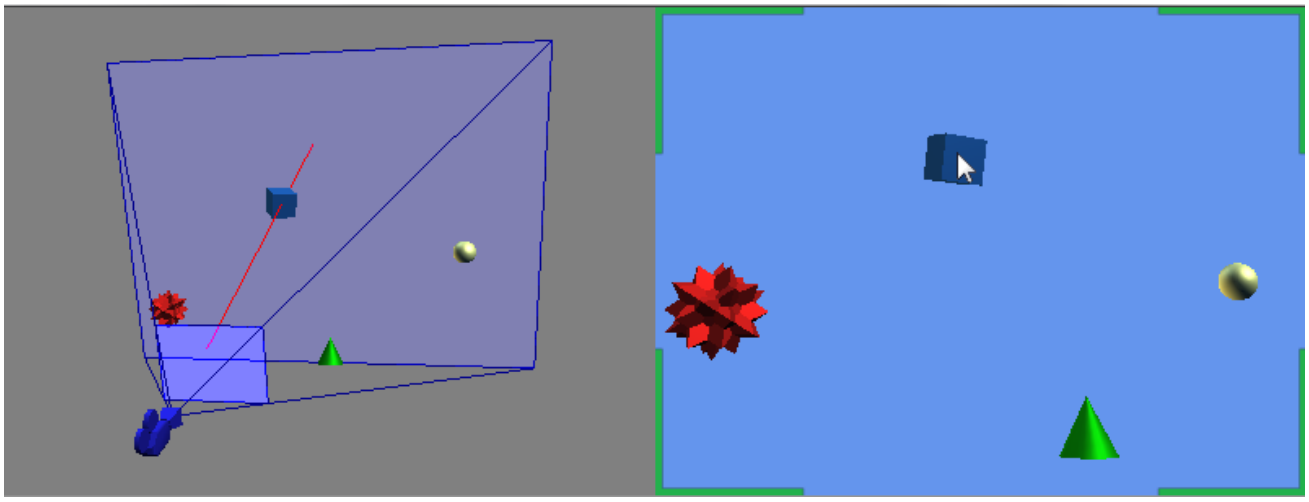
- When call rollDie() function, new ϕ and new θ is randomly generated. Each frame, current ϕ and θ are interpolated (linear interpolation).

5. Piece Class (child class of Model)

- At the corner of road, piece will move 90^0 to the left or right. So we divide the road from current position to target position to separate parts. Each part is a line, so points between start and end point are easily interpolated.
- Height of piece when jumping is calculate by formula:

6. Mouse clicking in 3D

- Game is currently used **glFrustum**.
- When user clicks, it will calculate the x, y coordinate in zNear plane.
- The pick vector is (x, y, -zNear), pick origin is (0, 0, 0). But it's using eye coordinate. To transform this eye coordinate pick ray into object coordinates, multiply it by the inverse of the ModelView matrix in use when the scene was rendered.
- Identify piece has been clicked by checking intersection of ray and piece's bounding box.



7. Dispatches user input event.

- **Game Scene**

- Left click: roll die, choose specify piece for moving.
- Right click: rotate camera, look around the board.
- Keyboard mapping:

S	Save screenshot
1 to 6	Cheating: get a specific number after roll die from 1 to 6
D	Hide/Display die
~	Toggle between full screen and windowed

- **Other Scenes**

- Left click: click to buttons.

8. Addition Library

- **Math library (matlib, collision)**

- Contain Math, Vector2, Vector3, Vector4, Matrix3, Matrix4, Matrix Stack and some functions like plus, minus, multiple, division, determinant, convert from degree to radian and reverse.
- Some type of bounding: bounding box, bounding sphere, bounding volume; Plane, Frustum, Ray and intersection function.

- **Simple OpenGL Image Library (SOIL)**

- A tiny C library used primarily for uploading textures into OpenGL. SOIL can also be used to save and load images in a variety of formats (useful for loading height maps, non-OpenGL applications, etc.)

II. Game Screenshot

1. Loading Scene

2. Main Menu Scene

3. Select Player Scene

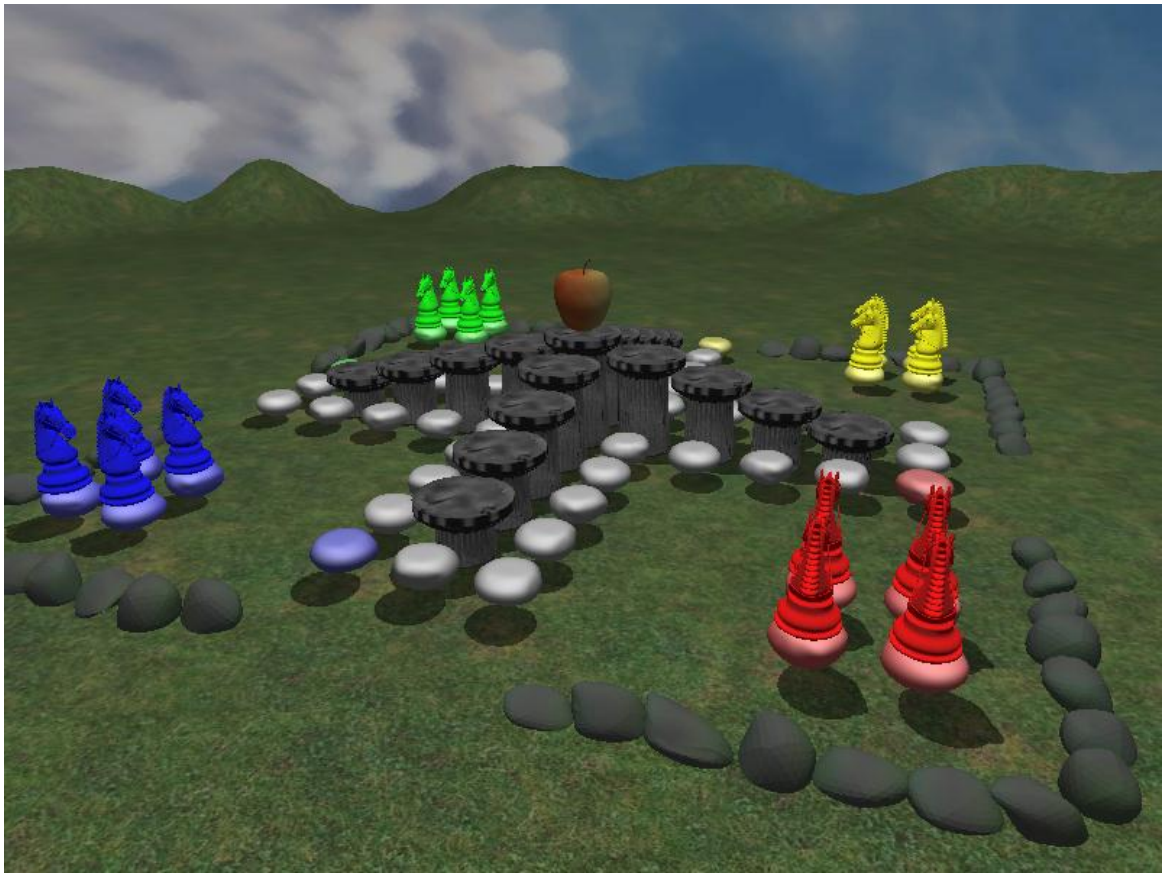
4. Option Scene

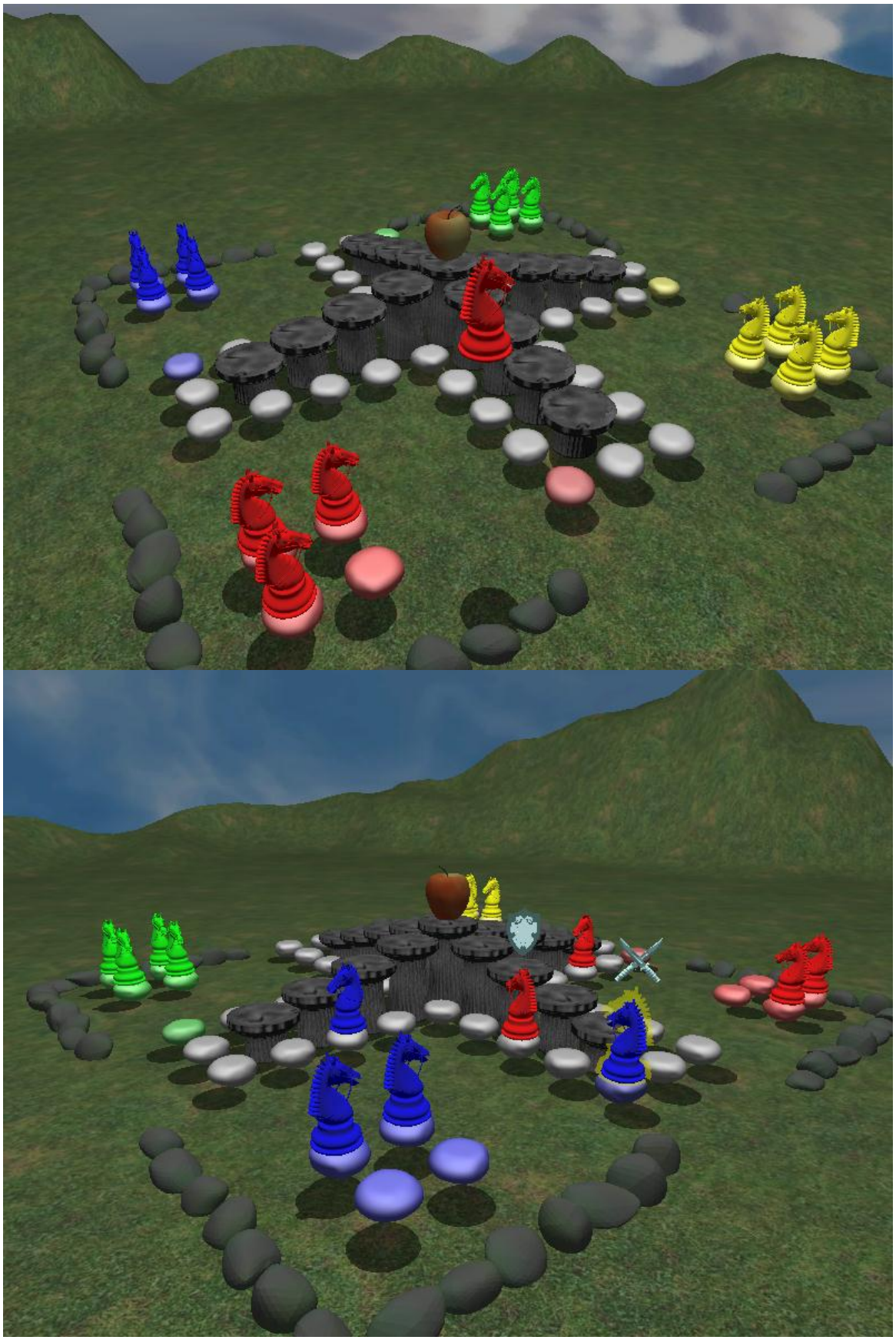
Not complete

5. About Scene

Not complete

6. Game Scene





III. Classes Overview

1. Math Library Classes

File: **mathlib.h, mathlib.cpp**

- **Math**: performing some common math operations such as convert from degree to radian and reverse, linear interpolation …
- **Vector2, Vector3, Vector4**: A 2, 3, 4-component vector class that represents a row vector and some operations.
- **Matrix3, Matrix4**: A row-major 3x3, 4x4 matrix class and some operations.

2. Collision Library Classes

- **BoundingBox**: A minimum box that contains the object inside it.
- **BoundingSphere**: A minimum sphere that contains the object inside it.
- **Plane**: performing a plane with a vector (a, b, c) and a number d

  a*x + b*y + c*z + d = 0
- **Ray**: an origin point and a direction vector. This class contains some intersection functions with BoundingBox, BoundingSphere…

3. Objects Classes

- **ModelOBJ**: parent of all classes in this group. Read data from obj and mtl file, process data and store it in memory.
- **Model**: child of ModelOBJ. This class contains function to get data from memory and draw object on screen with some parameter such as object's position, anchor point…
- **Piece**: Piece object and Piece function. There are some addition features of piece: shadow, highlight…
- **Die**: Die object, draw and roll die function.
- **Board:** Board object.
- **Rock:** Rocks create the road that Piece move on it.
- **Sky:** simulate the sky by drawing a sphere and add texture on it.
- **Sword & Shield:** Visual effect of piece's attacking.

4. 2D UI Classes

- **Sprite2D**: Store 2D texture and draw it in a rectangle in 2D or 3D view.
- **Button**: A button with 4 states: normal, hover, press, disable.
- **ToggleButton**: Toggle button with 3 states: on, off, disable.
- **RadioGroup**: A group of toggle button that if one button is on, others will be off.
- **ProgressTimerSprite2D**: Another type of Sprite2D, allow to draw only a part of texture on screen with specific percent.

5. Scene Classes

- **Scene**: Interface class. All other scenes inherit this class.
- **LoadingScene**: This scene displays the progress of loading data to memory.
- **MainMenuScene**: A menu scene, 4 buttons: Start, Option, About, Quits.
- **SelectPlayerScene**: Choose number of player, type of players in game (Human, Computer)
- **OptionScene**: Some options in game. (Not complete)
- **AboutScene**: Game's information, creator's information. (Not complete)

- **GameScene**: Game scene, user play game in this scene.

IV. Future Work

- Complete option scene, about scene.
- Add Pause button in Game, a tiny menu appear after press pause button, allow quitting current game without shutting down the application.
- AI for game, players can play with computer.
- Update the sound.
- Add Vietnamese rules.
- Add more Pieces' model.
- Model animation.