2023 Final Project Description

1. Basic Requirements

1. Choose one of the given reference topics or propose your own, but it must be a game type and use either Cocos2d-x or Unreal Game Engine library.
2. Complete the project individually or as a team, cross-team collaborations are allowed. If selecting a topic from the reference options, the team size should follow the limitations specified in the topic description. If choosing a self-proposed topic, the team size should not exceed 4 members.
3. Version control using Git is mandatory. The project must be publicly hosted on one of the designated Git hosting websites (see appendix). Please adhere to Git best practices, preserving individual commit records, which will be used to evaluate individual workloads.
4. Implement basic functionalities that can be compiled into an executable file and run without crashing.
5. No plagiarism is allowed, including copying code from open-source projects. However, the use of open-source libraries is permitted. If including the source code of an open-source library, it should be placed in a directory separate from your own source code.
6. Cocos2d-x must be used as the game development framework, and the project should not be limited to a console interface. This project emphasizes code and architectural design, so please avoid focusing excessively on visual aesthetics.
7. Utilize at least three specified C++ features (see appendix) in a consistent and reasonable manner.
8. Adhere to all specified code formatting requirements (see appendix).
9. Prior to **December 1st**, submit group information (including student ID and names), team leader (for team projects only), topic selection (or detailed description of self-proposed topic), and project Git repository address (can start with an empty repository containing only a ReadMe) to the teaching assistant (both teaching assistants should be contacted, contact information can be found under "Teaching Assistants and Contact Information").
10. Final project defense is required (content and defense schedule will be notified separately).
11. Submit the project source code, executable program, and documentation (specific documentation requirements will be announced separately) by the end of the semester.

2. Important Instructions

1. This project aims to assess students' ability in object-oriented program design, with a focus on understanding concepts such as classes, inheritance, polymorphism, and the utilization of complex C++ libraries.
2. The theme of this project is gaming. Due to time constraints, we do not impose strict requirements on the aesthetics of the game interface, game difficulty, or balance. However, the project should include a basic interface and reasonable game difficulty.
3. Regarding code plagiarism, students are allowed to reference code from open-source projects. However, it is expected that students complete the program writing based on their understanding of the referenced code, and provide proper attribution in the documentation by including the reference links to the code.
4. Regarding code plagiarism, we will conduct a plagiarism check on the project, with a focus on class design and game logic code. The code duplication rate should not exceed 30%. If any instances of code plagiarism are detected, appropriate actions will be taken as per the cheating policy.

Grading criteria

Grading will be conducted on a team basis, and members within the same team will generally receive similar final project scores. However, team members who contribute very little may have up to 40% of their final project score deducted. The contribution level will be assessed based on the git commit history.

Evaluation criteria

Evaluation Item	**Weight**
Level of implementation of basic features[*]	40% × Weight Score
Innovation of newly developed features beyond basic requirements	5%
Technical difficulty of newly developed features beyond basic requirements	15%
Workload of newly developed features beyond basic requirements	10%
Utilization of C++ features and functionalities	15%
Presentation during defense	10%
Documentation	5%
Completion with fewer than recommended team members[**]	Up to additional 15%
Bonus Points	Up to additional 30%

[*] The weight score of the chosen topic (specified in this document, see "Reference Topics") will affect your final score in that category. For example, with a weight score of 0.6, your final score in that category will not exceed 40% × 0.6 = 24%. Therefore, the maximum score you can achieve for the final project is 84 (out of 100). When the weight score is greater than 1, the additional points will be considered as bonus points (not conflicting with the bonus points in the evaluation criteria).

[**] Each reference topic will suggest a recommended team size. If the actual team size is smaller than the recommended size, the project will receive an additional 5% bonus for each team member less than the recommended size, up to a maximum of 15% bonus.

3. Reference Topics

Each topic is assigned a weight score, which represents the difficulty and workload of the project and also determines the upper limit of your final project score. A higher weight score indicates greater difficulty and expected workload for completing the project, resulting in a higher score for completing the same number of basic functionalities. Please choose a project that suits your level appropriately, to avoid receiving zero score due to lack of progress in a high-difficulty project, or being unable to obtain full marks due to choosing a project with too low difficulty.

Here, we provide a general outline of the required functionalities, and the specific details of the game should be designed by yourselves. The provided descriptions of basic and optional functionalities are not mandatory; for basic functionalities, you should implement similar functionalities, and for optional functionalities, we only provide partial references. You are encouraged to unleash your creativity and design additional interesting features or mechanisms.

2048

Weight Score: 0.7

Team Size Limit: <=1 person

Recommended Team Size: 1 person

Game Description: Players need to slide the screen to merge numbers and ultimately fill the board to score points.

Basic Features:

- **Basic Tiles**: 2 and 4, randomly appearing in empty spaces on the board.
- **Merge Rules**: When tiles of the same number collide during a slide, they merge into a new tile with a value equal to the sum of the original tiles.
- **Scoring**: Each time a new tile is created through merging, the player earns points equal to the value of the new tile.
- **Background Music**: Support background music to create a pleasant gaming atmosphere.
- **Basic Interface**: Includes a start screen, pause and resume functionality, and exit feature.
- **Leaderboard**: Support local leaderboard.

Optional Features (Welcome for original ideas):

- **Leaderboard**: Support online leaderboard.
- **Sound Effects**: Provide sounds when tiles are merged.

Carrot Fantasy

Weight Score: 1.2

Team Size Limit: <=3 people

Recommended Team Size: 3 people

Game Description: Reference game: "Defend the Radish"

Basic Features:

- Support at least 3 types of defense towers and the ability to remove them.
- Each defense tower should have attack effects, with at least projectile trajectories implemented.
- Display the health values of monsters and the radish.
- Support resource functionality, which can be used to purchase defense towers and can be obtained by killing monsters.
- Support at least 2 upgrades for each defense tower.
- Support at least 2 upgrades for the radish.
- Include at least 3 types of monsters.
- Include at least 2 maps.
- Support background music.
- Include a level selection screen and the ability to save progress (i.e., completed levels, available levels, and locked levels).

Optional Features (Welcome for original ideas):

- Support multiplayer online mode.
- Include sound effects for attacks, construction, and monster kills.
- Include the ability to select attack targets.
- Include a feature for destroying elements in the scene before constructing defense tower bases (Note: This feature is best implemented with the ability to select attack targets).
- Include special skills such as AOE/single-target damage skills or buff skills.
- Include a pause game feature.
- Include the ability to save the current progress when exiting the game and continue from the last checkpoint when re-entering the same level.
- Design boss monsters with special skills or high health points.

Teamfight Tactics

Weight Score: 1.6

Team Size Limit: <=5 people

Recommended Team Size: 5 people

Game Description: Reference games: "Dota Auto Chess," "Teamfight Tactics," "Battle of Golden Shovels"

Basic Features:

- Include initial interface and settings interface.
- Support background sound effects.
- Support multiple types of cards.
- Support card upgrading functionality.
- Support movement of the Tiny hero.
- Display health bars (red and blue) for cards on the field, with the blue bar filling up to enable skill activation.
- Support room creation and joining functionality.
- Include practice mode, allowing players to play against N AI players, where N >= 2.
- Include online mode, allowing players to play against N human players, with support for at least 2-player online battles.

Optional Features (Welcome for original ideas/Reference to original games):

- Support strengthening of various alliances (Challenge: requiring fulfillment of more card types).
- Include an equipment system (including equipment removal, randomization, etc.), and support equipment selection rounds.
- Support enhanced rune system.
- Include battle sound effects.
- Allow the Tiny hero to spectate other ongoing matches.
- Include a 2-player cooperative mode.

---

4. Bonus Features

Bonus Points: These points can be accumulated. Please inform the teaching assistant about the items to be checked for bonus points before the presentation.

1. Version Control and Team Collaboration
    1. Proper use of Git for version control and open-sourcing the project on GitHub.
    2. Fair and reasonable division of tasks among team members.
2. Code Quality and Security
    1. Use of unit testing to ensure code quality.
    2. Proper handling of exceptions.
3. Functionality and Architecture
    1. Beautiful user interface.
    2. **Extra Bonus**: Ability to port or directly run the game on Android or iOS platforms for participation in competitions.
    3. Clear project directory structure.
4. Others

1. No memory leaks.
2. The game runs without crashes when tested by the teaching assistant.
3. Maximum utilization of C++11 features.
4. Any other exceptional aspects that you would like to inform the teaching assistant about via private message.

5. Required Criteria

The following C++ features must be used, with a minimum of three:

- STL containers
- Iterators
- Classes and polymorphism
- Templates
- Exceptions
- Function and operator overloading
- C++11 or newer features

Your project must meet the following criteria to be accepted; otherwise, a 50% reduction in score will be applied:

- Consistent code formatting, and naming conventions must adhere to our predefined guidelines (such as Google C++ Style).
- Correct usage of C++ style type casting (`static_cast`, `dynamic_cast`) instead of traditional C-style type casting.
- Maximum utilization of the `const` modifier whenever possible.
- Appropriate design and implementation of comments.

6. Teaching Assistant Contact Information

- Wang Zeju
  - WeChat: Major-333
  - E-mail: wangzeju333@gmail.com
- Zhang Jipeng
  - QQ：914856774
  - E-mail：914856774@qq.com

7. References

Cocos2d-x：https://docs.cocos.com/cocos2d-x/manual/zh/

Git：https://git-scm.com/book/zh/v2

GitHub：https://www.jianshu.com/p/be9f0484af9d