GD/CCPROG3 Machine Project Specifications
1st Term, AY 2025-2026

I. Introduction

Chip's Challenge is a classic tile-based puzzle game first released in 1989, where players guide the main character, Chip McCallahan, through 149 increasingly difficult levels to earn his place in the elite Bit Busters Club. Each level is presented as a grid filled with computer chips to collect, locked doors, hazards like water, fire, and ice, as well as enemies with predictable movement patterns. To progress, Chip must gather the required chips, solve spatial puzzles, and reach the exit before the timer runs out. Keys unlock colored doors, boots provide protection against specific dangers, and pushable blocks help build bridges or block threats. Teleporters, force floors, and cloning machines add extra layers of complexity, requiring both logical thinking and precise movement. The game blends strategy, memory, and dexterity, making it a beloved challenge for puzzle enthusiasts.

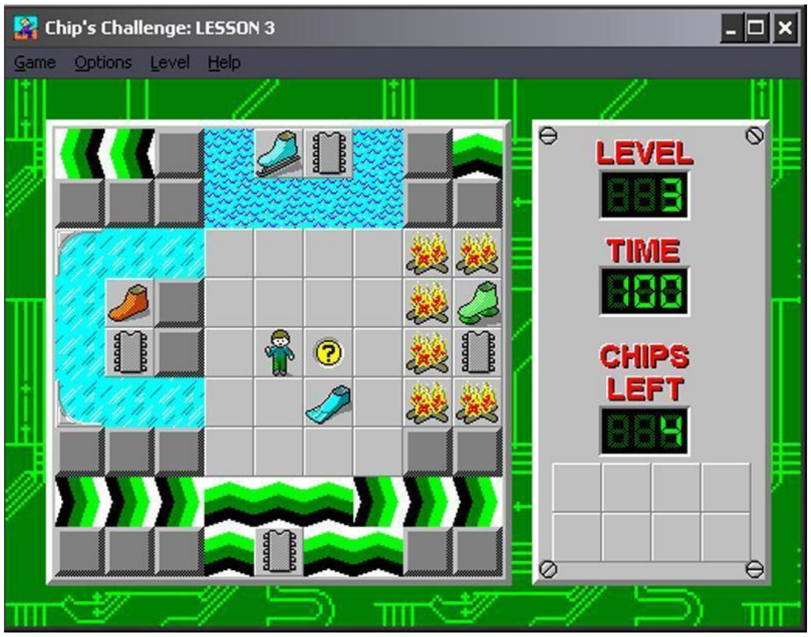

Image grabbed from: https://archive.org/details/chips_challenge_windows_3.x

II. Primary Objects

MCO1 Objects

Your task is to implement a toned-down version of Chip's Challenge with the following features:

1. Map
   a. The grid-based game world where Chip moves around and interacts with tiles, obstacles, and items.
2. Chip
   a. The player's character. Chip moves tile by tile, collecting chips and items, avoiding hazards, and finding the exit.
3. Tiles
   a. **Blank Tile** – An empty floor space where Chip can freely walk.
   b. **Water Tile** – A hazard. Chip drowns if he steps here without **Flippers**.

c. **Fire Tile** – Another hazard. Chip burns if he steps here without **Fire Boots**.
d. **Wall Tile** – A solid block. Chip cannot pass through.
e. **Exit Tile** – The goal of each level. Chip must collect all required chips before entering to proceed.
f. **Force Floor -** A tile that automatically pushes Chip in a fixed direction (up, down, left, or right) as soon as he steps onto it. Chip keeps sliding in that direction until he reaches a tile that stops him (like a wall, door, or block).

4. Doors
    a. **Red Door** – Requires a **Red Key** to open
    b. **Blue Door** – Requires a **Blue Key** to open
5. Inventory
    a. Collect Microchip - Microchips needed to complete each level. Some levels require a specific number before the exit opens.
    b. Collect Item
        i. **Red Key** – Unlocks one **Red Door**. Each key works once.
        ii. **Blue Key** – Unlocks one **Blue Door**. Each key works once.
        iii. **Flippers** – Lets Chip swim across **Water Tiles** safely.
        iv. **Fire Boots** – Lets Chip walk on **Fire Tiles** safely.
6. Next Level
    a. Triggered by stepping onto the **Exit Tile** once all chips are collected and requirements are met.

<u>MCO 2 Objects</u>
For this phase of the project you will be implementing an improved version of the game.

1. New Tiles
    a. **Ice Tile** - A slippery surface that makes Chip slide in the direction he was moving when he entered. Chip keeps sliding in a straight line until he either hits a non-ice tile, a wall, or some other obstacle. Unlike Floor Floor, Ice doesn't force a specific direction; it only continues Chip's last movement.
    b. A new element tile of your choice and a corresponding item
2. Naive Enemy
    a. Moves in a straight line, either left–right or up–down, and reverses direction when blocked by a wall or obstacle. Has no awareness of Chip's position — it just keeps pacing back and forth (or up and down). If Chip touches the enemy (or the enemy touches Chip), Chip loses a life.

III. Minimum Requirements

<u>MCO1</u>

You are expected to fully implement the objects mentioned above, along with at least two playable maps.

You may use special characters to represent the different tiles, objects, and characters in your map. A graphical user interface is not expected for this milestone.

MCO2

In addition to the MCO1 requirements, you are expected to fully implement the additional objects mentioned. A graphical user interface is expected for this milestone. You are not required to create your own sprites for this milestone you may use existing sprites.

References:
https://freebie.games/games/chips-challenge/play/
https://store.steampowered.com/app/346850/Chips_Challenge_1/
(Sprites)
https://www.reddit.com/r/chipschallenge/comments/1bofqqa/cc2s_win_tileset_but_paletteaccurate_and_with/#lightbox

IV. Milestones (see Deliverables Section)

a. MCO1 – due 9:00 pm, October 24, 2025 (F)

1. UML Class Diagrams for the model layer.
2. Implementation of the class the model layer.
3. Driver to test the basic application. No GUI display is expected. However, displays must be made in the Console.

b. MCO2 – due 12:00 nn, November 24, 2026 (M)

1. UML Class Diagrams (Object-Oriented)
2. GUI with Mouse-Controlled Inputs.
3. Complete implementation of the application
4. Implementation following the SOLID design principles.

V. Deliverables

The deliverables for both MCOs include:

1. The design and implementation of the solution should:
    - Conform to the specifications described above
    - Exhibit proper object-based / object-oriented concepts, like encapsulation and information-hiding, etc.
    - **NOT** be derived or influenced from any use of Generative AI tools or applications
2. Signed declaration of original work (declaration of sources and citations may also be placed here)
    - See **Appendix A** for an example
3. Softcopy of the class diagram following UML notations (in pdf or png)
    - Kindly ensure that the diagram is easy to read and well structured
4. Javadoc-generated documentation for proponent-defined classes with pertinent information
5. Zip file containing the source code with proper internal documentation
    - The program must be written in Java
    - Include external libraries that were used (e.g., JavaFX)
6. Test script following the format indicated in **Appendix B**
    - In general, there should be at least 3 categories (as indicated in the description) of test cases per method (except for setters and getters).
    - There is no need to test user-defined methods which are ONLY for screen design (i.e., no computations/processing; just print/println).
7. For MCO1 only: A video demonstration of your program

   - While groups have the freedom to conduct their demonstration, a demo script will be provided closer to the due date to help with showing the expected functionalities.
   - The demonstration should also quickly explain key aspects of the program's design found in the group's class diagram
   - Please keep the demo as concise as possible and refrain from adding unnecessary information
8. Groups should back-up their projects. A softcopy of the final unmodified files (for each phase) should be sent to their own emails, apart from regular submissions of progress in AnimoSpace and/or Git.

VI. Submission

All deliverables for the MCO are to be submitted via AnimoSpace. Submissions made in other venues will not be accepted. Please also make sure to take note of the deadlines specified on AnimoSpace. Late submissions will not be accepted.

VII. Grading

For grading of the MCO, please refer to the MCO rubrics indicated in the syllabus.

VIII. Collaboration and Academic Honesty

This project is meant to be developed as a pair for both phases (i.e., MCO1 and MCO2) barring any reports of freeloading or decision by the pair to split. In exceptional cases, a student may be allowed by their instructor to work on the project alone; however, permission should be sought as collaboration is a key component of the learning experience. Under no circumstance will a group be allowed to work on the MCO with more than 2 members.

A student cannot discuss or ask about design or implementation with other persons, with the exception of the teacher and their groupmate. Questions about the project specifications should be raised in the Discussion page in AnimoSpace. Copying other people's work and/or working in collaboration with other teams are not allowed and are punishable by a grade of 0.0 for the entire subject and a case may be filed with the Discipline Office. In short, do not risk it; the consequences are not worth the reward. Comply with the policies on collaboration and AI usage as discussed in the syllabus.

IX. Documentation and Coding Standards

Do not forget to include internal documentation (comments) in your code. At the very least, there should be an introductory comment and a comment before every class and every method. This will be used later to generate the required External Documentation for your project via javadoc. You may use an IDE or the appropriate command-based instructions to create the documentation, but it must be PROPERLY constructed.

Please note that you're not expected to provide comments for each and every line of code. A well-documented program also implies that coding standards are adhered to in such a way that they aid in the documentation of the code. Comments should be considered for more complex logic.

X. Bonus Points

Bonus points will only be awarded for MCO2. The above description of the program is the basic requirement. Any additional feature will be left to the creativity of the group. Bonus points would be awarded depending on the additional implemented features. These additional features could include:

- An intelligent enemy (e.g., one that gets closer and closer to Chip)
- Giving Chip the ability to shoot enemies

Depending on the scale of the new feature, additional points will be awarded to the group. However, make sure that all the minimum requirements are completely and correctly met first; if this is not the case then no additional points will be credited despite the additional features. To encourage the usage of version control, please note that a small portion of the bonus points for MCO2 will be the usage of version control. Please consider using version control as early as MCO1 to help with group collaboration.

XI. Resources and Citations

All sources should have proper citations. Citations should be written using the APA format. Examples of APA-formatted citations can be seen in the References section of the syllabus. You're encouraged to use the declaration of original work document as the document to place the citations.

Further, this is to emphasize that you DO NOT need to create your own sprites (background pictures, plant/zombie pictures, etc.) for the game. You can just use what's available on the Internet and just include these into your project, just make sure to cite your sources.

XII. Demo

Demo for MCO1 is via a video submission. All members are expected to be present in the video demonstration and should have relatively equal parts in terms of the discussion. Any student who is not present during the demo will receive a zero for the phase.

In MCO2, demo is live and will include an individual demo problem. Schedule for the demo will generally be during class time, but there may be other schedules opened by the faculty in case there is not enough time to accommodate everyone. Sequence (of who goes first in the class to do the demo) is determined by the faculty. Thus, do not be absent or late during announced demo days. A student or a group who is not present during the demo or who cannot answer questions regarding the design and implementation of the submitted project convincingly will incur a grade of 0 for that project phase.

During the demo, it is expected that the program can be compiled successfully in the command prompt and will run. If the program does not run, the grade for that phase is 0. However, a running program with complete features may not necessarily get full credit, as implementation (i.e., code) will still be checked.

XIII. Other Notes

You are also required to create and use methods and classes whenever possible. Make sure to use Object-Based (for MCO1) and Object-Oriented (for MCO2) Programming concepts properly. No brute force solution.

Statements and methods not taught in class can be used in the implementation. However, these are left for the student to learn on his or her own.

Appendix A. Template for Declaration of Original Work

Declaration of Original Work

We/I, [Your Name(s)] of section [section], declare that the code, resources, and documents that we submitted for the [1st/2nd] phase of the major course output (MCO) for CCPROG3 are our own work and effort. We take full responsibility for the submission and understand the repercussions of committing academic dishonesty, as stated in the DLSU Student Handbook. We affirm that we have not used any unauthorized assistance or unfair means in completing this project.

[*In case your project uses resources, like images, that were not created by your group.*] We acknowledge the following external sources or references used in the development of this project:

1. Author. Year. Title. Publisher. Link.
2. Author. Year. Title. Publisher. Link.
3. Author. Year. Title. Publisher. Link.

By signing this declaration, we affirm the authenticity and originality of our work.

Signature and date	*Signature and date*
Student 1 Name ID number	Student 2 Name ID number

[*Note to students:*

1. *Do not submit documents where your signatures are easily accessible. Ideally, submit a flattened PDF to add a layer of security for your digital signatures*
2. *You may use the eSignature feature of Google Docs. See this template.*]

Appendix B. Example of Test Script Format

Class: MyClass						
Method	**#**	**Test Description**	**Sample Input Data**	**Expected Output**	**Actual Output**	**P/F**
isPositive	**1**	**Determines that a positive whole number is positive**	**74**	**true**	**true**	**P**
	2	**Determines that a positive floating point number is positive**	**6.112**	**true**	**true**	**P**
	3	**Determines that a negative whole number is not positive**	**-871**	**false**	**false**	**P**
	4	**Determines that a negative floating point number is not positive**	**-0.0067**	**false**	**false**	**P**
	5	**Determines that 0 is not positive**	**0**	**false**	**false**	**P**