Name:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

RPG Maker MV Workbook

Contents

[Introduction 3](#_Toc97288830)

[Game Initialization Event 6](https://liveycstech-my.sharepoint.com/personal/csmith_ytech_edu/Documents/9th/2122/Challenges.docx#_Toc97288831)

[Maploader Event 7](https://liveycstech-my.sharepoint.com/personal/csmith_ytech_edu/Documents/9th/2122/Challenges.docx#_Toc97288832)

[Power Bracelet 9](https://liveycstech-my.sharepoint.com/personal/csmith_ytech_edu/Documents/9th/2122/Challenges.docx#_Toc97288833)

[Combo Lever (Any Order) 10](#_Toc97288834)

[Combo Lever (Specific Order) 12](#_Toc97288835)

[Reputation System 13](https://liveycstech-my.sharepoint.com/personal/csmith_ytech_edu/Documents/9th/2122/Challenges.docx#_Toc97288836)

[Magic Feather Jump 14](#_Toc97288837)

[Fishing Spot 16](#_Toc97288838)

[Fishing Skill and Rods 17](#_Toc97288839)

[Fishing Varieties 18](#_Toc97288840)

[Well 19](#_Toc97288841)

[Time System 20](#_Toc97288842)

[Day/Night Cycle 22](https://liveycstech-my.sharepoint.com/personal/csmith_ytech_edu/Documents/9th/2122/Challenges.docx#_Toc97288843)

[Dungeon/Boss Keys 24](#_Toc97288844)

[Butterdog Sniff Spots 26](https://liveycstech-my.sharepoint.com/personal/csmith_ytech_edu/Documents/9th/2122/Challenges.docx#_Toc97288845)

[Market Stall Theft *(solutions by Russel FB)* 27](https://liveycstech-my.sharepoint.com/personal/csmith_ytech_edu/Documents/9th/2122/Challenges.docx#_Toc97288846)

[Pickpocketing 29](https://liveycstech-my.sharepoint.com/personal/csmith_ytech_edu/Documents/9th/2122/Challenges.docx#_Toc97288847)

[Butterdog Sniff Spell 30](#_Toc97288848)

[Dad Joke Generator 33](#_Toc97288849)

# Introduction

*The instructor will go over these in class with you. These will give you the basic tools you need to complete the rest of the challenges in this booklet.*

* Editing a map
* Creating a New Map
* Quick Chest/Teleport/Door/Transfer
* Manual Transfer
* Door w/ Animation
* Remote Door w/ Animation
* Manual Chest
* Database Basics
* Event Editing and Pages
* Event Commands
* Event Movement/Animation
* Kid who runs in circles
* Manual Monster Fight
* Random Encounters
* Debug Room
* Item to TP to Debug Room
* Power Bracelet
* Butterdog

# Flow Chart Shapes

The function of a flowchart is determined by the shapes in it. There are five major shapes that are used in this workbook.

# Flow Chart Shape “Color Code”

This workbook also uses shades of black to signify different things. White shapes are instructions for the programmer, whereas black shapes are instructions for the program.

White shapes are instructions for you, the programmer:



Black shapes are Command Events. The Event will be named in the “Start” ellipse.

Gray/Faded shapes tell you that you need to insert code into an existing event. For example, if we have a flow chart like the one below:…and a different flow chart tells you to insert a new step, like this:…then the orginal flow chart will look like this when you’ve inserted the new steps:

# Game Initialization Event

All Switches start at ‘OFF’ and Variables at 0. We’ll create an event that that kicks off the game by setting all of our Switches and Variables to the correct values.

Create the following game object on the same map as your **Player Starting Position**:

 Create: Map Event

|  |  |  |
| --- | --- | --- |
| No Image | Name | Map Init |
| Page | 1 |
| Trigger | Autorun |

 Create: Map Event

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No Image | Name | Map Init | Self-Switch | A |
| Page | 2 |
| Trigger | Action |

Follow the flowchart below to complete the challenge:



Note that Page 2 of “Map Init” will always be blank. This is so the the first page will run once, and the event will never run again. Self-Switches are “persistent”, so even if you come back to this map later, it will not run again.

As you complete more and more challenges, some of them will tell you to insert a Switch or Variable change by inserting it into the “Init Map Event”. This means injecting that code where you see the trapezoid shape in the above flow chart.

# Maploader Event

We will create an event that we can copy/paste onto each map that will automatically change all of the game settings related to that map when we transfer in.

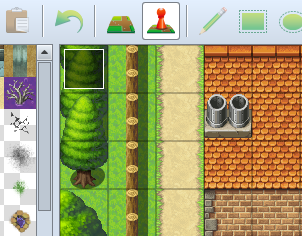
Start by creating the following game objects:

 Create: Map Event

|  |  |  |
| --- | --- | --- |
| No Image | Name | Maploader |
| Page | 1 |
| Trigger | Parallel |

Follow the flowchart below to complete the challenge:



This code seems sparse, but as you complete more challenges, you will be instructed to insert additional Command Events where the trapezoid is located in the flowchart above.

Note that we “Erase Event” instead of having a second page that requires Self-Switch A. This ensures that this event will respawn and run each time we enter the map.

Put the event in the upper left corner so that you’ll always be able to easily find it.

# Power Bracelet

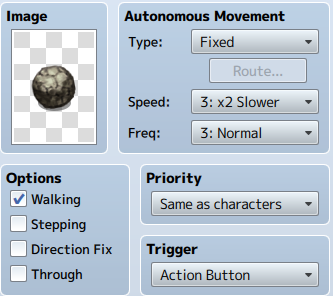
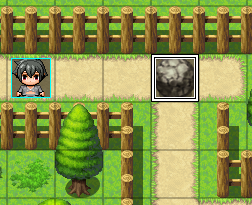
Design a classic game mechanic to push “boulders” that block our path until we’ve obtained the magic item “Power Bracelet”.

Create: Item

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 145 | Name | Power Bracelet | Type | Key Item |
| Consum. | No | Scope | None |
| Occasion | Never | Effect | None |
|  |  |  |  |  |

 Create: Map Event

|  |  |  |
| --- | --- | --- |
|  | Name | Boulder |
| Priority | Same |
| Trigger | Action |



If this is your first time using a Decision Step (Diamond), be sure to explore the “Conditional Branch” command event in its entirety. You do not need to edit the Event’s “Conditions” section.

When complete, you can copy/paste the boulder event around your game.

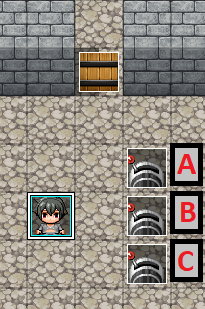
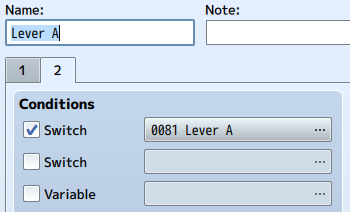


|  |  |
| --- | --- |
| This boulder will walk/move one step *away* from wherever the Player is standing when they use the Action button while possessing the Power Bracelet. Do not remove, teleport, or move the boulder in a fixed direction. | When testing your Event:   * Did you follow the flow chart *precisely*? * If you push the boulder against the wall, does it freeze the game? * The player does not need to provide any input after the initial Action Button press. |
|  |

# Combo Lever (Any Order)

Design a door that remains closed until three separate levers are pulled. Once all three are pulled, the door automatically opens.

1. Create three Switches, one for each lever (A, B, and C)
2. Create three lever events (A, B, and C). Each one’s first page has the lever facing left and takes no conditions. The second page of each will have the level facing right and the conditions require the associated Switch (A, B, or C) to be ON.
3. Create a Switch for the Door Lock
4. Create a Door event. The first page takes no conditions, and the image is a closed door. The second page requires the Door Switch to be on in the Conditions, and has no image.





# Combo Lever (Specific Order)

Design a door that remains closed until three levers have been pulled in a specific order. Pulling them in the wrong order resets the entire combination.

1. Follow all instructions for **Combo Lever (Any Order)**
2. Instead of using the flow chart in **Combo Lever (Any Order)**, use the flow chart below
   1. Remember, *white shapes* mean it’s something you as the programmer do while creating the actual program in *dark shapes*.



# Reputation System

Design a reputation system that increases a player’s “reputation” with townsfolk in the game for doing “good” deeds and lowers their reputation when they do “bad”. things.

 Create: Memory

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Variables | Reputation | 25 | Rep. Default | 25 | Rep. Bad | 10 |
| Reputation | Rep. Neutral | 20 | Rep. Good | 30 | Rep. Max | 50 |

Reputation Scores

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Evil | Bad | Neutral | Good | Revered |
| 0-9 | 10-19 | 20-29 | 30-39 | 40-50 |

 Create: Map Event

|  |  |  |
| --- | --- | --- |
|  | Name | Gossip |
| Page | 1 |
| Trigger | Action |

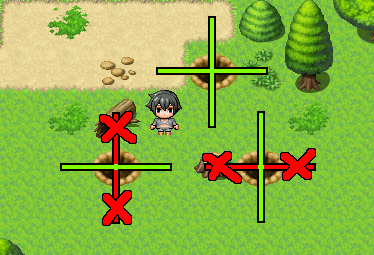
Remember the trapezoid in the flow chart from “Game Initialization Event”? That is where you insert the command events from the flow chart to the right. Variables whose values are set once and never change are known as “constants”. As we test our game, if we decide we want to change the values of our reputation system, we don’t have to track down every instance of them in the game. We just change the “constant” once. The Gossip NPC demonstrates this (flowchart below). Other challenges will use this system as well.



# Magic Feather Jump

Design a hole in the ground that can only be jumped over if the party owns a “Magic Feather” item. Make sure they cannot jump over something if the other side is blocked by an impassable tile.





# Fishing Spot

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 260 | Name | Small Fish | Type | Regular Item |
| Consumable | Yes | Scope | 1 Ally |
| Scope | 1 Ally | Effect | Recover 50 HP |

Design a fishing spot that goes over a body of water. Activating this spot gives you a random chance of catching a fish.

1. Create a “Fishing Spot” event over a body of water (must be Priority “Same as characters”)
   1. If you do not want all water to be fishable, add a water image to the event to signify this is a designated spot.
   2. If you want all water to be fishable, copy and paste the invisible event to all places you can fish after it is coded
2. Create a “Small Fish” item that replenishes a small amount of HP once.
3. Create a Variable called “Fishing Chance”





# Fishing Skill and Rods

Expand upon **Fishing Spot** by adding a Fishing Rod item and a skill that will grow over time.



New Item

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 260 | Name | Small Fish | Type | Regular Item |
| Consum. | Yes | Scope | 1 Ally |
| Scope | 1 Ally | Effect | Recover 50 HP |
|  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CREATE** | **Switches** | *N/A* | **Variables** | *Fishing Skill* |
|  | *Fishing Modifier* |



# Fishing Varieties

Expand upon **Fishing Spot** and **Day/Night Cycle** by adding larger fish, and fish that can only be caught at night.



# Well

Create a well which your player can draw water from.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 228 | Name | Water | Type | Regular Item |
| Consum. | Yes | Scope | 1 Ally |
| Scope | 1 Ally | Effect | Recover 20 MP |

 Create: Map Event

|  |  |  |
| --- | --- | --- |
|  | Name | Well |
| Priority | Same |
| Trigger | Action |





# Fishing Spot

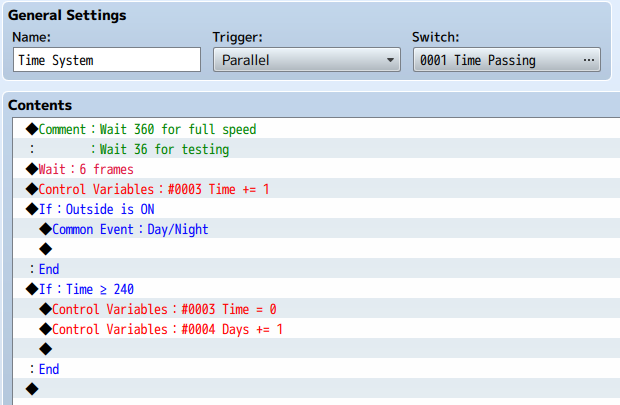
Design a fishing spot that goes over a body of water. Activating this spot gives you a fish.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 228 | Name | Water | Type | Regular Item |
| Consum. | Yes | Scope | 1 Ally |
| Scope | 1 Ally | Effect | Recover 20 MP |



# Time System

Design a system that can track in-game time in minutes/hours/days.



Because in-game time is different from real time, the numbers we use to represent it will be different as well. Our system will use 24 hours in a day, but only 10 “minutes” in an hour. Each in-game “minute” will last a chosen number of frames. *Remember to start the time by turning on “Time Passing” in your game.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CREATE** | **Switches** | *Time Passing* | **Variables** | *Time* |
|  | *Days* |



Create the Memory and flowchart:

# Day/Night Cycle

Expand upon **Time System** by making it adjust the color of the screen depending on the time of day if you are outside.

 Create: Memory

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Switches | Outside | Variables |  | N/A |



Insert the following into the flowchart for **Time System**:



You will need to turn “Outside” ON or OFF in your Map Loader Event

# 

# Dungeon Keys / Boss Key

Every classic dungeon has a series of doors that can be opened with one-time-use keys found throughout the dungeon. The “Boss” monster has a special key for themselves.

For this challenge, you will create two kinds of doors and two kinds of keys. A ‘Small Key’ opens all ‘Dungeon Doors’, and a ‘Boss Key’ opens the ‘Boss Door’

 Create: Map Event

|  |  |  |
| --- | --- | --- |
|  | Name | Dungeon Door |
| Page | 1 |
| Conditions | None |

 Create: Map Event

|  |  |  |
| --- | --- | --- |
|  | Name | Dungeon Door |
| Page | 2 |
| Conditions | Self-Switch A |

Create: Item

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 195 | Name | Boss Key | Type | Key Item |
| Consum. | No | Scope | None |
| Occasion | Never | Effect | None |
|  |  |  |  |  |

 Create: Map Event

|  |  |  |
| --- | --- | --- |
|  | Name | Boss Door |
| Page | 1 |
| Conditions | None |

 Create: Map Event

|  |  |  |
| --- | --- | --- |
|  | Name | Boss Door |
| Page | 2 |
| Conditions | Self-Switch A |

Create: Item

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 195 | Name | Small Key | Type | Key Item |
| Consum. | No | Scope | None |
| Occasion | Never | Effect | None |
|  |  |  |  |  |



The second page of each Door event will be left totally blank. This is the easiest way to allow the player to walk freely though the doorway once it’s been unlocked. Because of this, we only need one flow chart for the first page of each Door. Because the function is identical between doors except for the key we need, we will recycle the flow chart for both types of key/door combination.



# Remote Door

Make a door that only opens when the player presses a button elsewhere.

 Create: Map Event

|  |  |  |
| --- | --- | --- |
|  | Name | Door Lever |
| Page | 1 |
| Conditions | None |

 Create: Map Event

|  |  |  |
| --- | --- | --- |
|  | Name | Door Lever |
| Page | 2 |
| Conditions | Switch: DoorLever |

 Create: Memory

|  |  |
| --- | --- |
| Switches | DoorLever |

 Create: Map Event

|  |  |  |
| --- | --- | --- |
|  | Name | Door Lever |
| Page | 1 |
| Conditions | None |

 Create: Map Event

|  |  |  |
| --- | --- | --- |
|  | Name | Door Lever |
| Page | 2 |
| Conditions | Switch: DoorLever |

# Butterdog Sniff Spots

Design spots around trees/boxes/bushes that the player can see if “Butterdog” is in their party. They can interact with them once to get a free item.

Start by creating the following game objects:

 Create: Map Event

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Name | Sniff Spot | Options | Stepping |
| Page | 1 |  |  |
| Trigger | Action |  |  |

 Create: Map Event

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No Image | Name | Sniff Spot | Self-Switch | A |
| Page | 2 |
| Trigger | Action |



With Butterdog

Without Butterdog

# Market Stall Theft *(solutions by Russel FB)*

Create a farmer’s market stall that we can steal from when the owner isn’t looking.

Start by creating the following game objects:

 Create: Map Event

|  |  |  |
| --- | --- | --- |
|  | Name | Stall |
| Page | 1 |
| Trigger | Action |

 Create: Map Event

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Name | Stall | Self-Switch | B |
| Page | 2 |
| Trigger | Parallel |

 Create: Map Event

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Name | Farmer | Auto Move | Custom |
| Page | 1 |
| Trigger | Action |  |

Follow the flowchart below to complete the challenge:



For example, if the owner is standing to the upper right of the stall, you need to check to see if he’s facing RIGHT and check if he’s facing UP. Both of those directions turn Self-Switch A on. Self-Switch A signals a successful steal. Self-Switch B causes the owner to wait to “restock” the missing merchandise.

This challenge uses one Variable named “Player Facing” and several “NPC Facing XX”. *XX* is the number of the NPC you are tracking with this Variable (see flowchart)

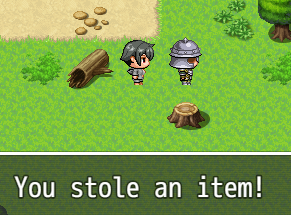
 Create: Memory

|  |  |
| --- | --- |
| Variables | Player Facing |
|  | NPC Facing XX |

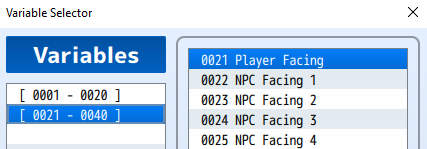
# Pickpocketing

Make some NPCs in your game that you can steal items from if you approach and interact with them from behind.

Follow the flowchart below:



From behind





Normal

For example, the first NPC you want to track will use a variable named “NPC Facing 1”. The second will use “NPC Facing 2” and so on.

# Butterdog Sniff Spell

Design a spell that can only be used by Butterdog that shows the HP of enemy monsters.



|  |  |
| --- | --- |
| Guard Arrest  Magic Mirror Transfer  Thirty-Second Town Portal  Slot Machine/Dice  One Step Floor Puzzle  Boulder Push Puzzle  Feather Jump Puzzle  Door Switch Puzzle  Name Password  Lens of Truth  Lost Woods  Boss Monster HP Formbar  Granny Storekeep  Lootbox  Ice Skate Puzzle  Boulder Ice Puzzle  Lockpicking  Torches  Minimap + Compass  One-way Ledge  Walking Monsters + Mimics  Running Boots  Fishing/Farming/Cooking/Thieving Skill  Bank  Sneaky Boots  Remote Door w/ Animation  Secret Money/Store  Remote Door Lock  Reputation Effects  Reputation Reset NPC  Lava Walking Boots  Hookshot  Cooking Spot  Clock  Farming with Seeds  Farming with Watering Can  Mimics  Magic toot  Jail Sentence  Sit/pass time  Bad rep bounty hunters  Password name |  |

# Dad Joke Generator

function addNewlines(str) {

var result = '';

while (str.length > 0) {

result += str.substring(0, 50) + '\n';

str = str.substring(50);} return result;}

function reqListener () {

console.log(this.responseText);

$gameMessage.add(addNewlines(this.responseText))

}

var xhr = new XMLHttpRequest();

xhr.addEventListener("load", reqListener);

xhr.open("GET", "https://icanhazdadjoke.com/");

xhr.setRequestHeader('Accept', 'text/plain')

xhr.send();

 Create: Actor

|  |  |  |
| --- | --- | --- |
| No Image | Name | Init |
| Other | This Actor totally blank! |
|  |

dasdadasdadadasdadsaddddddddddddddddddddddddddddddddddddd

◆Text：None, Window, Bottom  
：Text：Would you like to fish?  
◆Show Choices：Yes, No (Window, Right, #1, #2)  
：When Yes   
 ◆Show Balloon Icon：Player, Exclamation (Wait)  
 ◆Text：None, Window, Bottom  
 ：Text：You caught a fish!  
 ◆  
：When No   
 ◆Text：None, Window, Bottom  
 ：Text：You're right. Fish murder is probably bad.  
 ◆  
：End