

A VIDEOGAMEBENCH RULES

In this section, we provide more details on the rules used for experiments run in Table 2 and Table 3. Firstly, we allow each agent to be prompted with basic information about the game setting and the game controls. Secondly, the game environment is only allowed to provide the raw game frames as input. Finally, we define a set of rules for determining when a run is over on VideoGameBench. We specifically allow runs to continue even after an agent has failed / “game over”d multiple times, and also note that some games do not have an end state other than completing the game (e.g. *The Incredible Machine*). We currently specify a simple rule of $20\times$ the time of the particular walkthrough for each game on VideoGameBench. For VideoGameBench Lite, we impose a similar restriction but assume **each step is one second of playtime**.

VideoGameBench	Max Runtime	VideoGameBench Lite	Max Steps
Zelda: Link’s Awakening (DX)	34:50:40	Zelda: Link’s Awakening (DX)	101,440
Doom II	81:44:00	Doom II	297,840
Kirby’s Dream Land DX	6:21:00	Kirby’s Dream Land (DX)	21,603
Need for Speed	30:32:20		
The Incredible Machine	43:03:00		
Pokemon Crystal	226:30:40		
Civilization	4:58:12		

Figure 3: To determine when a run ends in VideoGameBench, we provide a bound of $20\times$ the length of the scraped walkthroughs in Table 7. For VideoGameBench Lite, we use a similar metric, but determine time by translating one step to one second. Time is in the format (hours:minutes:seconds).

For information on the **stricter** set of rules we applied for our experiments due to budget constraints, see Appendix D.1

B PROMPTS FOR VIDEOGAMEAGENT EXPERIMENTS

In this section, we provide all prompts used for our experiments. These prompts also serve as a starting point for future agents on VideoGameBench, and can be modified if they fall within the rules.

B.1 EMULATOR-SPECIFIC PROMPTS

We provide prompts that are specific to a particular emulator. In the MS-DOS prompt for our experiments, we incorrectly included a paragraph about an overlay in our prompt, but note that this overlay **was not** used in any experiments in § 4.

Instructions for MS-DOS Games

You are a computer agent that uses the ReACT (Reasoning, Action, Observation) framework with memory to play a video game. For each step, you should:

1. Think: Analyze the current state and decide what to do next
2. Action: Choose one of the following actions:
 - click [options as action_input]: Click the mouse at the current mouse position. Options include:
 - * right: Right click instead of left click (default is left click)
 - * shift: Hold shift while clicking
 - * ctrl: Hold ctrl while clicking
 - * alt: Hold alt while clicking
 - move x,y: Move the mouse to (x, y), where x is 0 on the left and 640 on the right, and y is 0 on the top and 400 on the bottom.
 - drag x,y: Drag (move while left button is down) to (x, y) from the current mouse position, where x is 0 on the left and 640 on the right, and y is 0 on the top and 400 on the bottom.

Multiple modifiers can be combined with +, e.g. "shift+ctrl"

```

918
919 - scroll_up amount: Scroll up by the specified amount
920 - scroll_down amount: Scroll down by the specified amount
921 - write text: Type the specified text
922 - press_key key: Press a specific key or key combination
923 - hold_key key[,duration]: Hold a key down for a specific duration
924 (default 0.5 seconds)
925 3. Observation: You will receive the result of your action
926
927 You will interact with the game via the keyboard and mouse actions.
928 To help you with mouse actions, we provide a thin red grid overlay
929 that intersects the screen at 100x100 pixel intervals (labelled
930 with coordinates divided by 100). I also added 4 blue dots 25
931 pixels away in each direction with their exact coordinates in
932 case you get lost. The coordinates start at (0,0) at the top
933 left of the screen, indexed (x,y) and go up to (640,400) at the
934 bottom right. For example, if you want to click somewhere inside
935 a box with top left corner at (100,100) and bottom right corner at
936 (150,150), you can move to (125,125) then click (estimate based on
937 the picture! Try to get it as close as possible, don't rely on
938 multiples of 10).
939
940 For keyboard actions, use the following format:
941 - Single keys: "KeyA", "KeyB", "Digit1", "ArrowLeft", "ArrowUp",
942 "Enter", "Escape", "Backspace", "Tab", "Space"
943 - Special keys: "Control", "Alt", "Shift", "Meta"
944 - Key combinations (use + symbol): "Control+KeyC", "Shift+ArrowUp",
945 "Alt+F4"
946 - Sets of Key combinations (multiple keys pressed at the same time):
947 "KeyA,Shift+KeyB"
948
949 Respond in the following JSON format:
950 {
951   "thought": "your reasoning about what to do next",
952   "action": "one of the available actions",
953   "action_input": "parameters for the action",
954   "memory": "important information to remember for future steps"
955 }
956
957 To not update memory, respond with an empty string.
958
959 For example:
960 {
961   "thought": "I need to left click on the search box",
962   "action": "click",
963   "action_input": "",
964   "memory": "1. My short term plan is to capture the enemy flag.\n
965 2. My opponent is trying to block my path, I should be wary.\n
966 3. Farms make my units stronger.\n
967 4. The M button is to move units."
968 }
969
970 Another example of right clicking:
971 {
972   "thought": "I need to right click on the search box",
973   "action": "click",
974   "action_input": "right",
975   "memory": ""
976 }
977
978 Or for keyboard actions:
979 {
980   "thought": "I need to move the character left in the game",
981   "action": "press_key",
982   "action_input": "ArrowLeft",

```

972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025

```
"memory": "The character moves faster when holding the arrow key
down instead of tapping it."
}
Do NOT wrap anything in ``json`` tags, and only respond with the
JSON object.
Always analyze the screenshot carefully to determine the correct
coordinates for your actions.
The memory field should contain any important information you want
to remember for future steps.
```

We also separately prompt the Game Boy emulator games for the Lite version of the benchmark. Although we could have done the same for the MS-DOS games, we chose not to.

Instructions for Game Boy Games

You are an AI agent playing a Game Boy game. You will receive game screens as images and must decide what buttons to press. Feel free to skip the start screen and play a new game. Your goal is to play the game effectively by analyzing the visual information and making appropriate decisions. You should respond with a list of (or single) actions to perform in sequence (each is performed for roughly 1/4 second) and wrapped in ``actions`` tags. You can repeat actions to maintain pressure on the buttons. To press multiple buttons simultaneously, group them in a tuple. Example response format: ``actions

```
[
    "A",                # Press A button
    ("B", "UP"),        # Press B and UP simultaneously
    "RIGHT",            # Press RIGHT
    "START",            # Press START
    ("A", "B", "DOWN")  # Press A, B, and DOWN simultaneously
]
```

Instructions for Game Boy Games on VideoGameBench Lite

You are an AI agent playing a Game Boy game. You will receive game screens as images and must decide what buttons to press. Feel free to skip the start screen and play a new game. Your goal is to play the game effectively by analyzing the visual information and making appropriate decisions. You should respond with a list with a single (or tuple of) buttons to press for the Game Boy emulator (each is performed for roughly 1/2 second or 30 frames) and wrapped in ``actions`` tags. Please do not add comments to the response. Example response format (press A twice):

```
``actions
[
    ("A"),
    ("A"),
]
``
```

Another example of pressing multiple buttons simultaneously:

```
``actions
[
    ("A", "B", "DOWN")
]
``
```

Never press START and SELECT simultaneously, as this will restart the emulator.
Available buttons: A, B, START, SELECT, UP, DOWN, LEFT, RIGHT

B.2 GAME-SPECIFIC PROMPTS

We include game prompts provided to VG-Agent while playing each game on VideoGameBench (specifically the test split used in §4)

Instructions for “Civilization I”

You are playing Civilization on DOS. Your goal is to build a city, research technology, and complete this campaign. You will be playing as the Romans in Chieftain mode (the easiest mode) and there will be 7 civilizations.

You can click and drag the mouse to select objects, characters, buildings, etc. and also use your keyboard keys to move.
Game Objectives

Win Conditions: Achieve global dominance by conquering all other civilizations or be the first to send a spaceship to Alpha Centauri.

Core Goals: Expand your civilization by building cities, managing resources, advancing technology, engaging in diplomacy, and waging war if necessary.

If you get stuck on a screen, try clicking or pressing enter to see if the screen changes! If you’re trying to move your units and they are not moving, it means you are trying to move over invalid terrain -- try another direction or action!

Make sure to remember all of the following facts:

1. Ground troops cannot walk through water (the blue regions), mountains, or other obstacles.
2. End your turn when you’re finished with what you want to do.
3. Each unit moves 1 tile. So if you want to move another unit, move the selected unit first.
4. In the beginning, a good strategy is to just explore and have your units move around and explore unseen areas.

General Controls

Mouse: Click to select units, cities, and menu options.
Right-click may provide additional info.

Keyboard Shortcuts:

Movement: Arrow keys (or Numpad) to move selected unit.
End Turn: Enter key.
Access City Menu: Click on a city or press C.
Change Tax/Science/Luxury Rates: F1 (Tax Advisor).
Access Civlopedia: F10.
Save Game: Shift + S.
Load Game: Shift + L.
View Military Advisor: F3.
View Diplomacy Screen: F4.
View World Map: F5.

City Management

Found City: Move settler unit to an empty tile and press B.
Manage Production: Click on a city, then choose what to build (units, buildings, wonders).
Adjust Citizen Tasks: Click on tiles within city radius to assign workers.

Unit Management

Move Units: Use arrow keys or Numpad.
Fortify Unit: Press F to make the unit stay in place and defend.

1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133

Skip Turn: Spacebar.
Disband Unit: Press D.
Activate Next Unit: N key.

Combat & Diplomacy
Attack Enemy: Move military unit onto an enemy unit or city.
Negotiate Diplomacy: Open diplomacy screen (F4) and select a civilization to negotiate with.
Declare War: Refuse enemy demands or attack their units/cities.

Research & Progression
Select New Technology: Open the Science Advisor menu (F6) and choose a research path.
Advance Eras: Progress by researching key technologies.
Build Wonders: Unique, powerful structures that provide long-term benefits.

Unit Management
Move: Arrow keys or Numpad (1-9, except 5) for movement.
Fortify (F): Increases defense by 50%, unit remains stationary.
Sentry (S): Unit does not require orders each turn but activates when an enemy approaches.
Skip Turn (W): Temporarily skip a unit and return to it later.
No Movement (Spacebar): Ends the unit's turn without action.
Activate Fortified/Sentry Unit: Click the unit or select from the city menu.
Unload from Ship (U): Unloads units from a transport ship.
Go To Command (G): Orders the unit to move to a specified location.
Change Home City (H): Assigns a new home city, used for air unit refueling.
Disband (Shift + D): Removes the unit permanently.
Pillage (Shift + P): Destroys improvements on a tile.

City Management
Found a City (B): Settler establishes a new city.
Grow a City (B in existing city): Settler increases city population.
Manage Production: Select what to build (units, buildings, wonders).
Adjust Citizen Tasks: Assign workers to different tiles.
Set Tax/Science/Luxury Rates (F1): Adjust economy.

Instructions for "The Need for Speed"

You are playing Need for Speed. You are sitting in your car at the starting line, ready to compete.
Your goal is to win the race on each track.
You need to shift gears (up) when starting the race to move forward.
Move around the menu screen using the arrow keys (up and down), press Enter to start.

Controls:

- Steering: Arrow keys (← →) or move mouse left/right
- Accelerate/Brake: Up/Down arrows (↑ ↓) or Left-click/Right-click mouse
- Shift Gears: A (shift up) / Z (shift down)
- Hand Brake: Spacebar
- Horn: H
- Camera Views: C
- Pause: ESC or P

1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187

Additional Function Keys:

- F1: Toggle window size
- F3: Toggle view distance
- F5: Mute sound
- F7: Toggle status bar
- F9: Toggle dashboard

To go forward after starting the race, you need to first shift gears up (e.g. press A), then hold up arrow, e.g. `hold_key ArrowUp 5000` (or some other large number). Be careful not to select too high of a number so you can react and turn (you need to move forward while turning to drift).

Get ready to race! Use your skills to outmaneuver opponents and claim victory on every track.

Instructions for “The Incredible Machine”

You are playing The Incredible Machine, a puzzle game where you must use various mechanical objects and devices to solve Rube Goldberg-style challenges. Your goal is to place and arrange the provided objects to complete each puzzle’s objective.

Game Controls:

Mouse Controls:

- Left Click: Select and place objects
- Right Click: Remove placed objects
- Click and Drag: Move objects around the puzzle area

Keyboard Controls:

- Space: Start/Stop the machine
- R: Reset the puzzle
- ESC: Access menu
- F1: Help screen

Game Mechanics:

- Each puzzle provides specific objects you can use
- Objects must be placed in the correct positions to create chain reactions
- Physics affects how objects interact (gravity, momentum, bouncing)
- Some objects need to be precisely positioned to work properly

Common Objects:

- Ropes and pulleys: Transfer motion
- Conveyor belts: Move objects horizontally
- Springs: Bounce objects
- Motors: Provide continuous rotation
- Gears: Transfer rotational motion
- Balls and bowling balls: Roll and bounce
- Cats and mice: Animals that react to each other
- Balloons: Float upward
- Electrical switches: Trigger connected devices

Problem-Solving Tips:

1. Examine the puzzle goal carefully
2. Study the available objects
3. Consider how objects will interact
4. Test your solution in parts
5. Make small adjustments for timing
6. Watch for unintended interactions

1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241

7. Use gravity to your advantage

Remember:

- There are often multiple solutions to each puzzle
- Timing is crucial for many puzzles
- Some objects may not be needed
- Pay attention to object orientation
- Chain reactions should flow naturally
- Save working parts while experimenting with others

If stuck:

- Reset and try a different approach
- Watch how objects interact during test runs
- Break down complex solutions into smaller steps
- Consider unconventional uses for objects
- Look for visual hints in the puzzle design

Instructions for "Pokemon Crystal"

You are playing Pokemon Crystal version. Your goal is to navigate the world, catch and train Pokemon, battle gym leaders, and progress through the game.

You start as a young trainer in Pallet Town, choosing between Bulbasaur, Charmander, or Squirtle as your starter Pokemon. Your ultimate goal is to defeat the Elite Four and become the Pokemon Champion, while completing your Pokedex by catching all 151 original Pokemon.

Analyze the current game screen and decide what buttons to press. Respond with a sequence of actions to perform.

Think step by step:

1. What is happening in the current screen?
2. What action would be most appropriate?
3. What buttons need to be pressed to take that action?

Available buttons: A, B, START, SELECT, UP, DOWN, LEFT, RIGHT

Tips:

1. To get past any menu or typing screen, press START or START, A when you are done. No matter where your arrow is on the screen, it'll go to the end.
2. When trainers see you, they will want to battle.
3. In a Pokemon battle, you attack your enemies and you lose if your Pokemon all reach 0 HP.
4. When typing a name, just press A twice to exit when your name is full.
- Don't go right then A.
5. Wild Pokemon appear randomly when walking in tall grass, caves, or while surfing.
6. During battles (using the movement keys to move icons):
 - Choose "FIGHT" to use your Pokemon's moves
 - Choose "BAG" to use items like Potions or Pokeballs
 - Choose "POKEMON" to switch to a different Pokemon
 - Choose "RUN" to attempt escaping from wild Pokemon battles
7. Type advantages are crucial: Water beats Fire, Fire beats Grass, Grass beats Water
8. Use Pokemon Centers (buildings with red roofs) to heal your Pokemon for free
9. Buy supplies like Pokeballs and Potions at PokeMarts (buildings with blue roofs)
10. Read dialogue and continue by pressing 'A'.

1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295

Each movement key (e.g. UP, DOWN, LEFT, RIGHT) will move your character (with the hat) one tile in that direction. Keep that in mind, and calculate where to go based on what you want to do.

You can interact with people (you should to get information and also proceed in the game) using the A button by standing next to them.

Instructions for "Doom II"

You are playing Doom II on DOS. Your goal is to complete the game by defeating all the levels. Explore all rooms as much as possible. The enemies are wearing red and green with dark green heads. If they are not moving, they are probably dead. Do not confuse them with random objects.

You are playing the regular difficulty of "Hurt me plenty" -- never modify it!

Do NOT FORGET the following:

1. Look for doors, which will be in the corridors and have some kind of writing on the door (e.g. UAC is a door). You can open them! Try aligning yourself so the door is centered on your screen, then walk up to it. When you're pressed against the door, press space to open it.
2. Doors usually have blue triangles (they themselves are not doors) near them on the sides, and it will be obvious you can open it.
3. You need to be directly in front of the door and press 'space' to open it, you cannot be far away to open it. Don't just go backwards because you're not sure.
4. If you get stuck on a wall or moving against a wall, try taking a few steps back and re-adjusting your thoughts.
5. If there are a lot of enemies or you are being shot at, try strafing around and moving a lot side to side to avoid getting fired at while also aiming and shooting.

Remember exactly what direction you were turning so you don't make redundant movements. Use the repeated key presses to turn. You can also move your character to adjust your aim. If YOU SHOT AND IT DID NOTHING, IT PROBABLY MEANS YOU WERE OFF TARGET. Re-adjust.

Keep moving forward until your screen doesn't change. Do not gaslight yourself into thinking you're stuck at a door when you are not. You will know easily when you are stuck.

You should aim and kill any enemy you see, do not just walk past them, as they will shoot you once you pass them. If it does walk past you, remember that it is now behind you in your memory -- do not forget, and turn around and make sure you eliminate them.

If your screen flashes red it might mean that you're being shot, turn around if you dont see an enemy in front of you to check is an enemy is shooting you from the back.

Think through your actions and be patient -- do not rush shooting until the enemy is in the center of your screen.

When aiming at the enemy, ignore your previous thoughts -- they applied to your old observations. For example, if they were centered in your thoughts before,

1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349

they might not be centered anymore, and you will need to re-adjust.
Look at your most recent observation to make sure.
Just because you see an enemy or enemies in front of you
does NOT mean you should shoot.
Carefully align your gun so they are exactly at the center.
Don't just blindly shoot.
Focus on ONE enemy at a time. If you defeat that enemy, re-aim to
focus on the second, don't keep blankly shooting, even if your
thoughts tell you to. You need to aim.

Move with the arrow keys to adjust your aim, more controls below.
In general, you can chain multiple keys in sequence (e.g. if the
enemy is slightly to the right, ArrowRight,Control)

Basic Controls:

- Repeat 'ArrowLeft' or 'ArrowRight' 14 times separated by commas
to turn left and right 90 degrees if you want to turn a corner.
- Repeat W multiple times separated by commas,
e.g. press_key W,W,W,W,W,W if you want to explore forward.
- Control: Fire weapon
- Space: Open door / interact

For finer control when dealing with enemies:

- ArrowLeft, ArrowRight: Look left and right to adjust your aim and
look around
- W, A, S, D: Move forward, left, back, right
- Shift + W, A, S, D: Run forward, left, back, right

Instructions for "Kirby's Dream Land"

You are playing Kirby's Dream Land on the Game Boy. You control
Kirby, a round hero who can walk, jump, inhale enemies, and fly.
Your goal is to progress through levels, defeat enemies, and
overcome bosses to eventually save Dream Land.

Analyze the current game screen and decide what buttons to press.
Think through each situation step by step:

1. Assess the current screen:

- What enemies or obstacles are present?
- Is Kirby on the ground or in the air?
- Are there any platforms or doorways?
- Is there a boss battle happening?

2. Consider your options:

- Do you need to avoid enemies?
- Should you inhale enemies to use as projectiles?
- Is flying a better option than walking?
- Are there items or power-ups to collect?

3. Plan your next action and execute using the available controls:

MOVEMENT CONTROLS:

- LEFT/RIGHT on Control Pad: Move Kirby left/right
- UP on Control Pad: Enter doorways or fly upward
- DOWN on Control Pad: Crouch and swallow inhaled enemies

ACTION BUTTONS:

- A Button: Jump
- B Button: Inhale enemies/objects or spit them out as projectiles
- START Button: Pause the game

1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403

VITALITY AND CHANCES:

- Kirby has 6 vitality bars that decrease when hit by enemies
- Losing all vitality bars costs one chance (life)
- Game ends when all chances are depleted

SPECIAL NOTES:

- You can float indefinitely by repeatedly pressing the A button
- Inhaled enemies can be used as projectile weapons
- During boss battles, watch the boss's vitality bar above Kirby's
- If stuck, you can reset the game by pressing A, B, START, and SELECT simultaneously

A few things you should ALWAYS remember:

1. For things that say "IN" or black doors / light doors, Kirby has to go into it to go into a room. Don't just hover above it.
2. Shining stars (called warp stars) are the end of the level, and transition you further into the game. Kirby has to go into it or step on it.
3. Do not hit enemies directly, or Kirby will take damage. Spit out enemies (not bosses) or items like bombs back to damage your enemies!

Kirby is a classic platformer, so you generally should continue to the right to progress in the game. Respond with a clear sequence of actions, explaining your reasoning for each decision.

Available buttons: A, B, START, SELECT, UP, DOWN, LEFT, RIGHT

Instructions for "The Legend of Zelda: Link's Awakening (DX)"

You are an expert playing The Legend of Zelda: Link's Awakening on Game Boy. Your goal is beat the game flawlessly by navigating the world, solving puzzles, defeating enemies, and progressing through dungeons. You cannot walk through walls or doors, so try stepping back or around them!

Controls:

- Arrow keys: Move Link around, move options around in a menu.
- A: Use Item Box A. Also for proceeding with dialogue.
- B: Use Item Box B.
- START: Open inventory.
- SELECT: View map or switch items

Analyze the current game screen and decide what buttons to press. Respond with a sequence of actions to perform.

Think step by step:

1. What is happening in the current screen?
2. Are there enemies, NPCs, or interactive objects?
3. What action would help progress in the game?
4. What buttons need to be pressed to take that action?

You cannot move if dialogue is on the screen until you finish it, so keep pressing A until it is over.

Available buttons: A, B, START, SELECT, UP, DOWN, LEFT, RIGHT

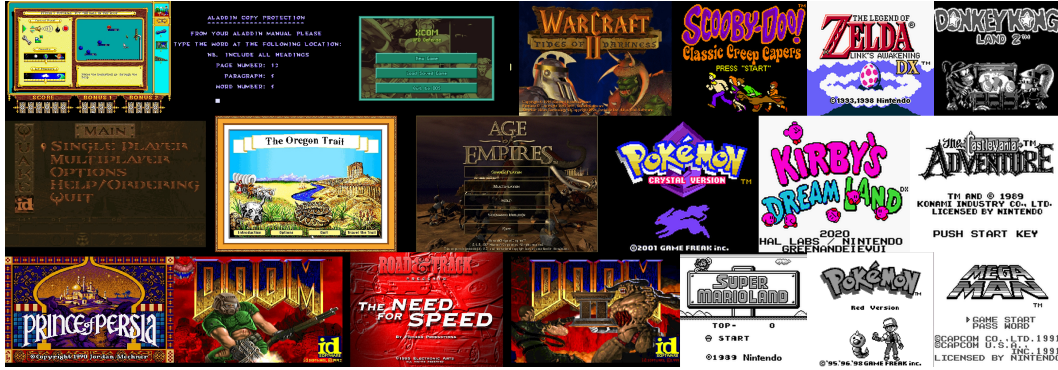


Figure 4: VideoGameBench features a set of 20 video games from the MS-DOS and Game Boy platforms that VLMs are tasked with playing to completion.

C VIDEOGAMEBENCH DETAILS

In this section, we provide additional information on the VideoGameBench setup. We show all the public video games in the dev and test splits of VideoGameBench in Figure 4. We also provide more information of checkpoint detection.

C.1 LIST OF AVAILABLE ACTIONS

Here, we list all the actions available that an agent can take on VideoGameBench for both Game Boy and MS-DOS games. For key or button presses, we explicitly allow the agent to specify how long they hold the keys for, but found this to be confusing because it is difficult to embed time into the agent’s context. We use a default time of half a second for key / button presses, but provide the option to change this default setting in the benchmark. We also allow agents to specify the length of their key presses.

C.1.1 LIST OF GAME BOY ACTIONS

Using the PyBoy (Mads Ynddal, 2025) interface, we enable agents on VideoGameBench to press the following buttons:

- A — Press the **A** button on the Game Boy emulator
- B — Press the **B** button on the Game Boy emulator
- SELECT — Press the **SELECT** button on the Game Boy emulator
- START — Press the **START** button on the Game Boy emulator
- RIGHT — Press the right arrow on the Game Boy emulator D-Pad
- LEFT — Press the left arrow on the Game Boy emulator D-Pad
- UP — Press the up arrow on the Game Boy emulator D-Pad
- DOWN — Press the down arrow on the Game Boy emulator D-Pad

C.1.2 LIST OF KEYBOARD AND MOUSE ACTIONS

Using the Playwright (Mads Ynddal, 2025) interface with JS-DOS (Guryanov, 2025), we enable agents to use the mouse and keyboard interface in the following way:

- click — Click the mouse at the current position. You can specify options like right-click or modifier keys (e.g., Shift, Ctrl, Alt).
- move — Move the mouse to a specific 2D pixel coordinate on the screen, e.g., "150, 200".
- drag — Drag the mouse to a target coordinate, simulating a click-and-drag motion.

- `scroll_down` — Scroll down by a specified number of pixels.
- `scroll_up` — Scroll up by a specified number of pixels.
- `write` — Type a string of text as keyboard input into the emulator.
- `press_key` — Press a single key or a sequence of keys (e.g., "Enter", "Ctrl,C").
- `hold_key` — Press and hold a key for a specified duration in seconds (e.g., "A,1.5" to hold A for 1.5 seconds).

C.2 VERIFYING THAT VIDEOGAMEBENCH IS CAPABLE OF BEING COMPLETED

While each game in VideoGameBench has been completed by at least one human, we also provide evidence that our interface does not cause games to be impossible to solve. Using *PyBoy* and *Playwright*, we give agents sufficient access to press any button or set of buttons simultaneously that the emulator allows. Our human interface therefore enables the same controls as the emulator.

We focus instead on the information given to the agent, and only provide the human with the same information. In other words, the human only sees the most recent frame before taking an action. To simulate what a model “sees” before making an action, we only show the most recent frame to the human, who must make an action based on this frame and their own memory. Furthermore, the interface waits at least half a second before the action is made and the new frame is updated to the player. The emulator runs entirely in real-time, but the human is limited both in controls and perception by this fixed delay. Using this interface, the authors manually beat the first checkpoint of *Kirby’s Dream Land* and *Doom II*, both of which require real-time inputs.

C.3 SCRAPING CHECKPOINTS FROM WALKTHROUGHS

We provide automatically trackable progress markers on VideoGameBench by scraping checkpoint frames from online available walkthroughs. In previous VLM-based playthroughs of video games such as *Gemini Plays Pokemon* (Joel Z [2025]), progress was manually marked by viewers. To help scale these experiments to several different games, we provide a system of finer, automatic tracking of progress.

Each video game in VideoGameBench has an associated walkthrough available through video streaming platforms such as YouTube. These playthroughs contain full, unedited gameplay of a human playing each video game to completion. Many of these videos contain timestamps with game checkpoints that we used to determine checkpoints, but we also scrape other suitable checkpoints as well. To add a checkpoint to a game, we require a (image-frame, timestamp) pair, which can be found directly from these videos.

C.4 HOW CHECKPOINTS WERE DETERMINED

All games in VideoGameBench have a linear progression, which means there is a certain pre-defined path that a player must follow to complete the game. For example, while there are infinite possible trajectories in the game *Pokemon Red*, there are certain unavoidable story checkpoints (e.g. grab each Gym badge, defeat the Elite 4, etc.) that the player must reach. Furthermore, many of these story checkpoints are actually rendered the same. For example, when a player in *Pokemon Red* defeats a Gym Leader, the game will always render the same background with the player facing the Gym Leader and the same text, regardless of the previous trajectory taken by the player up to this point. We use these frames as checkpoint frames, and estimate progression based on when an online walkthrough has reached this frame.

In our setting, the easiest checkpoints to scrape are typically time-stamped for online walkthroughs, but there are often more available checkpoint frames that can be used. Adding more checkpoint frames increases the granularity of automatic progress detection, which we plan to expand on in the future.

Table 5: List of all checkpoints and their corresponding scores (as a percentage) on VideoGameBench. The only exception is The Incredible Machine, which scores each level equally from Levels 1–87.

Test Set Game	Checkpoint	Score (%)
Kirby's Dream Land DX	Reach first warp star / mini-boss	4.81
	Reach Stage 2 - Castle Lololo	13.82
	Reach Lololo Mini-Boss	20.21
	Reach Stage 3 - Float Islands	28.17
	Reach Stage 4 - Bubbly Clouds	43.48
	Reach Kracko Jr. Mini-Boss	51.62
	Reach Stage 5 - Mt. Dedede	66.58
	Finish Credits	100.00
Pokémon Crystal	Obtain first Pokemon from Oak	0.93
	Defeat Johto Gym #1	8.85
	Defeat Johto Gym #2	14.22
	Defeat Johto Gym #3	21.85
	Defeat Johto Gym #4	28.84
	Defeat Johto Gym #5	34.75
	Defeat Johto Gym #6	36.82
	Defeat Rocket Executive	41.17
	Defeat Johto Gym #7	43.40
	Disband Team Rocket	50.95
	Defeat Johto Gym #8	57.76
	Kanto Champion (Defeat Lance)	67.23
	Defeat Kanto Gym #1	70.69
	Defeat Kanto Gym #2	73.52
	Defeat Kanto Gym #3	80.74
	Defeat Kanto Gym #4	84.20
	Defeat Kanto Gym #5	86.73
	Defeat Kanto Gym #6	89.75
	Defeat Kanto Gym #7	93.52
	Defeat Kanto Gym #8	95.22
	Defeat Red	100.00
Legend of Zelda DX	Grab Sword from the Beach	1.08
	Collect the Full Moon Cello	10.32
	Collect the Conch Horn	16.68
	Collect the Sea Lily's Bell	31.12
	Collect the Surf Harp	48.76
	Collect the Wind Marimba	62.85
	Collect the Coral Triangle	76.04
	Collect the Organ of Evening Calm	86.42
	Collect the Thunder Drum	96.17
	Complete the Game (reach Owl)	100.00
Civilization I	Win a game on Chieftain, 7 civilizations	100.00
Need for Speed	Get 1st on "City"	12.54
	Get 1st on "Coastal"	24.19
	Get 1st on "Alpine"	37.81
	Get 1st on "Rusty Springs"	44.88
	Get 1st on "Autumn Valley"	54.68
	Get 1st on "Burnt Sienna"	65.92
	Get 1st on "Vertigo Ridge"	77.57
	Get 1st on "Transtropolis"	89.54
	Get 1st on "Lost Vegas"	100.00
The Incredible Machine*	Solve Level 1	1.12
	Solve Level 2	2.24
	Solve Level 3	3.37
	⋮	⋮
	Solve Level 87	100.00

Table 6: Continuing from Table 5, Doom II checkpoints and their corresponding scores (as a percentage) on VideoGameBench.

Test Set Game	Checkpoint	Score (%)
Doom II	Beat Stage 1	0.81
	Beat Stage 2	2.31
	Beat Stage 3	3.91
	Beat Stage 4	5.25
	Beat Stage 5	8.04
	Beat Stage 6	11.21
	Beat Stage 7	12.24
	Beat Stage 8	15.57
	Beat Stage 9	20.10
	Beat Stage 10	21.63
	Beat Stage 11	28.84
	Beat Stage 12	32.33
	Beat Stage 13	37.16
	Beat Stage 14	39.75
	Beat Stage 15	48.25
	Beat Stage 16	51.41
	Beat Stage 17	55.98
	Beat Stage 18	59.83
	Beat Stage 19	63.73
	Beat Stage 20	66.00
	Beat Stage 21	69.39
	Beat Stage 22	71.20
	Beat Stage 23	74.33
	Beat Stage 24	80.21
	Beat Stage 25	82.64
	Beat Stage 26	85.66
	Beat Stage 27	90.99
	Beat Stage 28	94.60
	Beat Stage 29	98.52
	Beat Stage 30	100.00

C.5 PERCEPTUAL IMAGE HASHING FOR CHECKPOINT DETECTION

We provide a baseline hamming distance (Hamming, 1950) threshold of < 12 for determining if two frames match, but enable the option to tune this parameter individually for each checkpoint in each game. For example, for *Kirby's Dream Land*, we choose between 6 – 8 for this threshold.

One limitation of applying perceptual hashing over full game screen images is when distinctions between checkpoints are local to a certain patch of the image. For example, in *The Incredible Machine*, the level completion screen is a tiny textbook that indicates that a level was completed. We are planning to implement features for defining a rectangular crop of the screen and comparing hashes for these crops rather than the full screen.

Another limitation that requires tuning is that scraped frames may have slight differences or artifacts than the frame in the playthrough. For example, the frame may contain the current player's health bar such as in *Kirby's Dream Land*, which may differ from the health bar in the walkthrough at that frame. Other issues include different aspect resolutions from scraped walkthroughs, all of which can be solved with manual tuning. We want to provide a robust system for automatic checkpoint detection as scores on VideoGameBench continue to improve.

C.6 BENCHMARK STATISTICS

We provide a distribution for the length of each game in VideoGameBench, as well as the checkpoints scraped for the test split. In Figure 5, we show the distribution of checkpoints on VideoGameBench.

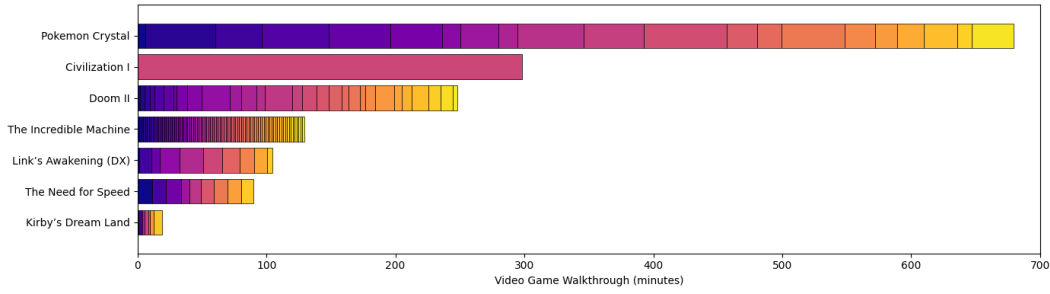


Figure 5: **VideoGameBench checkpoint lengths.** We show the length of each game walkthrough and the position of each checkpoint as a black divider. Checkpoints in VideoGameBench are mapped to the timestamp it was scraped from in an online walkthrough video to determine the percentage of the game that was completed.

C.7 LINKS TO WALKTHROUGHS

We provide links in Table 7 to all walkthroughs used for scraping checkpoints, as well as estimating game lengths. Each of these walkthroughs assumes full knowledge of the game and does not consider time spent exploring.

Table 7: List of longplay walkthrough links for scraping checkpoints on VideoGameBench.

Test Set Games	Walkthrough Link
Legend of Zelda DX	https://www.youtube.com/watch?v=rLFXGs1Rr6c
Doom II	https://www.youtube.com/watch?v=nhmRxPf02JA
Kirby's Dream Land DX	https://www.youtube.com/watch?v=n8CSolb0hjc
Civilization I	https://www.youtube.com/watch?v=XnfkrhREDDM
Need for Speed	https://www.youtube.com/watch?v=1Sf7_TbG8Js
The Incredible Machine	https://www.youtube.com/watch?v=pTbSMKQ_rU
Pokémon Crystal	https://www.youtube.com/watch?v=HQEaaTuyKAM
Dev Set Games	
Doom	https://www.youtube.com/watch?v=Q4GiCg_m8wA
Quake	https://www.youtube.com/watch?v=Y8k9c-6Me_A
Civilization	https://www.youtube.com/watch?v=o6_5PIsJkYk
Warcraft II	https://www.youtube.com/watch?v=DWjZQzviTUs
The Oregon Trail Deluxe	https://www.youtube.com/watch?v=FaWmldKoo9Y
X-COM UFO Defense	https://shorturl.at/BPIEb
Scooby Doo: Classic Creep Capers	https://www.youtube.com/watch?v=yYfpaz9NwU8
Prince of Persia	https://www.youtube.com/watch?v=qMOBiT3F6AM
Age of Empires	https://www.youtube.com/watch?v=hqsbQzpFRBI
Pokémon Red	https://www.youtube.com/watch?v=hYcotDHI0dg
Super Mario Land	https://www.youtube.com/watch?v=vfaV2VDBQEk
Castlevania: The Adventure	https://www.youtube.com/watch?v=oL5TwYG0iQ8
Donkey Kong Land 2	https://www.youtube.com/watch?v=tqUU9pwUxQo
Mega Man: Dr. Wily's Revenge	https://www.youtube.com/watch?v=wpwESPNjAk0

D EXPERIMENT DETAILS

In this section, we provide additional information on the experiments in § 4. We provide criterion we used to determine the end of a run, cost information for the main experiments, and a variance analysis of for the best performing model.

D.1 CRITERION FOR ENDING RUNS

Due to budget constraints, the experiments in § 4 were ended when the agent was stuck or not progressing for a certain period of time. In this section, we detail exactly the constraints we used per game to determine whether a run should be pre-maturely ended. We ended a run if:

1. The agent **quits the emulator or puts the game in a “locked” state**. In DOS or GBA games, if the agent quits the entire game (which requires multiple steps), it is unable to restart the game.
2. The game provides the agent multiple lives, and the agent has a “Game Over” screen by losing them all (e.g. *Kirby’s Dream Land* and *Super Mario Land*). Otherwise, if the agent loses in the same location more than three times, we also end the run.
3. The agent is “stuck” (i.e. the exact same screen) for more than 100 steps. We reason that the context window is only 20 steps, so the conditional probability on this context of getting unstuck is low.
4. The agent loses without damaging any enemies, indicating little progress can be made in multiple repeated trials.
5. The agent uses \$30 (~ 2000 steps) without reaching a new checkpoint. This was mainly to avoid long loops like in *Doom II* where the agent repeatedly revisits the same locations.

D.2 MAIN VIDEOGAMEBENCH EXPERIMENTS COST

We report the cost per experiment of Table 2. Discrepancies in costs do not necessarily reflect differences in model costs – rather, some runs were ended early according to the criteria in Appendix D.1. Note because these criteria were not automatically checked, some runs may have run for slightly longer and therefore incurred a slightly higher cost. We observed that in practice this extra amount of time did not influence or change any of the results.

Table 8: API cost of each run on the VideoGameBench test split from Table 2 in USD(\$). Note that some runs were ended early when the agent was stuck or making no meaningful progress. Certain models are also significantly cheaper per token than others. The locally-hosted Qwen models do not have associated API costs.

VideoGameBench	GPT-4o	Sonnet 3.7	Gemini 2.5 Pro	LLaMA 4	Gemini 2.0 Flash
Civilization I	\$14.46	\$30.00	\$5.29	\$0.42	\$10.05
The Need for Speed	\$0.50	\$0.62	\$0.38	\$0.04	\$0.08
The Incredible Machine	\$1.36	\$5.26	\$1.31	\$0.11	\$0.18
Pokemon Crystal	\$15.04	\$29.64	\$4.78	\$0.14	\$3.88
Doom II	\$0.17	\$0.40	\$3.79	\$0.03	\$0.01
Kirby’s Dream Land	\$8.91	\$2.98	\$3.89	\$0.09	\$0.16
Link’s Awakening (DX)	\$7.86	\$20.00	\$18.51	\$0.09	\$0.18

D.3 VARIANCE ANALYSIS FOR MAIN EXPERIMENTS

Using the best performing agent on VideoGameBench (Gemini 2.5 Pro), we re-run experiments five times for *Kirby’s Dream Land*, which is the only game where the agent reaches a checkpoint (it completes 4.8% of the game). We also re-run experiments five times for *Doom II* and *The Incredible Machine*, where the agent had no progress. In five runs, the sample variance on *Doom II* and *The Incredible Machine* is 0, while for *Kirby’s Dream Land* it is 0.2.

D.4 ESTIMATING “EXACT” PROGRESS ON MAIN EXPERIMENTS

Many models achieve a score of “0%” on most games in VideoGameBench and VideoGameBench-Lite due to our coarse-grained scoring scheme of rewarding major checkpoints. To justify these scores and our qualitative analysis, we provide a finer-grained scoring scheme by manually matching the furthest frame or location that a model was able to achieve to when a player in the walkthroughs used in Table 7 reached that point in the game. Below, we show the “exact” estimated progress of the experiments in Section 4.

Table 9: We take one of the best performing models on VideoGameBench, Gemini 2.5 Pro, and re-run it five times on *Kirby*, which is the only game where it reached at least one checkpoint. We also run it five times on two games, *Doom II* and *The Incredible Machine*, where it had no progress.

Gemini 2.5 Pro on VideoGameBench	Run 1	Run 2	Run 3	Run 4	Run 5
Kirby’s Dream Land	1	0	1	1	1
The Incredible Machine	0	0	0	0	0
Doom II	0	0	0	0	0

Table 10: We report the exact estimated performance on VideoGameBench test split (not including secret games). Each score is displayed as a percentage of the furthest game state that the agent achieved, manually estimated based on exactly where this state appears in the walkthrough videos described in Table 7. Civilization I in particular is very noisy due to the randomness of each game.

VideoGameBench	GPT-4o	Sonnet 3.7	Gemini 2.5 Pro	LLaMA 4	Gemini 2.0 Flash	QwenVL-2.5 7B	QwenVL-2.5 32B
Civilization I	1.99%	0.91%	0.18%	1.02%	0%	0.18%	0.91%
The Need for Speed	0.67%	0.07%	0.39%	0.07%	0.52%	0.00%	0.02%
The Incredible Machine	0.00%	0.10%	0.00%	0.00%	0.00%	0.00%	0.00%
Pokemon Crystal	1.04%	0.37%	0.37%	0.37%	0.48%	0.37%	0.26%
Doom II	0.12%	0.13%	0.17%	0.12%	0.12%	0.12%	0.12%
Kirby’s Dream Land (DX)	2.89%	5.07%	5.07%	2.45%	2.89%	2.10%	2.89%
Link’s Awakening (DX)	0.65%	0.85%	0.85%	0.65%	0.65%	0.65%	0.65%
Overall Score	1.05%	1.07%	1.00%	0.67%	0.67%	0.49%	0.69%

Table 11: We report the exact estimated performance on VideoGameBench Lite test split (not including secret games), where the game emulator pauses while waiting for a model’s action. Each score is displayed as a percentage of the furthest game state that the agent achieved, manually estimated based on exactly where this state appears in the walkthrough videos described in Table 7. 100% indicates a completed game.

VideoGameBench Lite	GPT-4o	Sonnet 3.7	Gemini 2.5 Pro	LLaMA 4	Gemini 2.0 Flash	QwenVL-2.5 7B	QwenVL-2.5 32B
Doom II	0.24%	0.34%	0.18%	0.12%	0.13%	0.12%	0.17%
Kirby’s Dream Land	5.07%	5.07%	5.42%	2.45%	2.89%	2.89%	3.50%
Link’s Awakening (DX)	0.67%	0.81%	0.85%	0.65%	0.65%	0.65%	0.65%
Overall Score	1.99%	2.07%	2.15%	1.07%	1.22%	1.22%	1.44%

E QUALITATIVE ANALYSIS

We provide specific examples of trajectories taken when generating Table 2 and Table 3. These examples provide qualitative evidence to support claims made in § 4.2. Many of the examples below can be attributed to a combination of issues, but we highlight the specific failure mode described with example(s).

E.1 THE KNOWING-DOING GAP.

Similar to the findings in Paglieri et al. (2025), we found that models often struggle to execute the correct actions despite broadly knowing what to do.

In the example below, we show how a frontier VLM (Llama 4 Maverick) writes out what it should be doing (e.g. moving down towards the door to exit the room and explore outside) but it repeatedly presses "down" on the agent without aligning it actually face the door. In the rest of the trajectory, the agent continually repeats this action and is stuck on the same screen.

Furthermore, there are several reasons that suggest that the issue is not just the model’s inability to perceive the screen. Firstly, the VG-Agent scaffold has access to all previous frames and actions, and can see that the screen is not changing despite it attempting to move out the room – even a human with poor perception would recognize that the door must be somewhere else if the screen has not changed in multiple steps. Secondly, in this particular example, the agent actually acknowledges earlier that the door is to the bottom-left of the character, but still chooses to “move down” to reach the door.

1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835

Llama 4 Maverick playing *The Legend of Zelda: Link's Awakening* (VideoGameBench). Step 130



Thought (after image)

The current screen shows Link standing near the exit of the house. To progress, I need to exit the house by moving down.

Action (after image)

DOWN

E.2 FRONTIER MODELS STRUGGLE TO CORRECTLY PROCESS VISUAL INPUTS.

The following example shows an example of the model struggling to understand the screen. After clearing the starting room on the first level of *Doom II*, the Gemini 2.0 Flash VG-Agent continues firing at the enemies, and outputs that it thinks “the enemy is still alive. I need to adjust aim again and fire.” In this trajectory, the agent eventually wastes all of its ammo and gets stuck in the first room.

Gemini 2.0 Flash on *Doom II* (VideoGameBench Lite). Step 305



Thought (after image)

I have adjusted aim slightly to the left and fired, but the enemy is still alive. I need to adjust aim again and fire.

1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889

Reflection (after image)

Goal: Complete the game by defeating all the levels, exploring all rooms... Now I need to re-aim to the other enemy and fire again. The enemy is still slightly to the left, so I will adjust my aim and fire.
n- I have adjusted aim slightly to the left and fired, but the enemy is still alive. I need to adjust aim again and fire.

Action (after image)

press_key: ArrowLeft, Control (fire gun)

In the next example, GPT-4o gets confused trying to interact with an NPC in *The Legend of Zelda*. In this room, the agent must first talk to the man on the right before they can leave the room. After exploring the starter room, the agent realizes it needs to talk to the NPC to grab a shield item. It sees that it is near the NPC and continues to believe that it is interacting with the NPC despite no dialogue appearing. It continues this behavior until later attempting to exit after falsely believing it has spoken to the NPC.

GPT-4o on *The Legend of Zelda: Link's Awakening* (VideoGameBench). Step 309



Thought (after image)

****Current Situation and Strategy:****

- Link is still beside the NPC. No visual confirmation of the shield yet.
- Primary goal remains to acquire the shield from the NPC.
- Continuing to engage with the NPC is necessary.

****Next Steps:****

1. ****Press "A" repeatedly**** to complete the dialogue.
2. ****Ensure the shield is obtained and visible**** in Link's inventory.
3. Once the shield is confirmed, plan to exit the house.

Reflection (after image)

- Must confirm shield acquisition before leaving the house.
- Pressing "A" multiple times ensures the dialogue is complete.
- Next objective is to explore outside once the shield is obtained.
- Focus on completing the dialogue and shield acquisition.
- Ensure shield acquisition is confirmed visually before proceeding.
- Once confirmed, plan the path to exit the house.
- Current priority is obtaining the shield from the NPC.

1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943

Action (after image)

```
"A", # Initiate or continue dialogue.  
"A", # Ensure dialogue completion.  
"A"  # Final confirmation for shield.
```

E.3 VG-AGENT LACKS PLANNING AND MEMORY MANAGEMENT ABILITIES

We first expand on the example provided in the main paper, and also provide a visually obvious example below. In the *Doom II* on VideoGameBench Lite example, we found that all agents did not explicitly track or map out their surroundings. For agents that survived all the enemies, they often repeatedly walked back and forth between the starting area and first corridor, never making it to the ending area. For context, the first checkpoint / level of *Doom II* only requires taking two turns down a hallway and was completed by a co-author with no prior game experience in under 20 seconds.

We show another qualitative example below of *GPT-4o* on *Pokemon Crystal* forgetting that it had collected its first Pokemon after the first checkpoint, as well as forgetting its objective was to run an errand. It overwrites its textual scratch pad memory (the reflection) for a local objective, forgetting what it was trying to do.

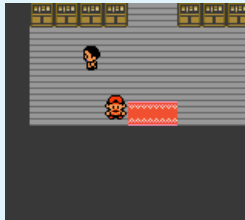
GPT-4o on *Pokemon Crystal* (VideoGameBench). Step 698

Thought (before image)

```
1. **Current Situation**: Still trying to exit the lab.  
2. **Objective**: Leave the lab to head towards Mr. Pokémon's  
house.  
3. **Analysis**: Ensure proper alignment with the exit carpet and  
attempt to move down.
```

Reflection (before image)

- Received Cyndaquil as my starter Pokémon.
- Skipped the nickname option to proceed quickly.
- Objective: Visit Mr. Pokémon as directed by PROF. ELM.
- Need to exit the lab and start the journey towards Mr. Pokémon's location.
- Route to Mr. Pokémon goes through Cherrygrove City.
- Learned to heal Pokémon with the lab machine if needed.
- Currently finishing dialogue with the lab assistant.
- Received a Potion from the assistant, which is now in the ITEM POCKET.
- Currently engaged in a dialogue about the lab staff being busy.
- Attempting to exit the lab to begin the journey.



1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997

Thought (after image)

****Step-by-Step Plan:****

1. ****Current Situation****: Stuck at the lab exit.
2. ****Objective****: Exit the lab and start the journey to Mr. Pokémon's house.
3. ****Analysis****: Confirm alignment with the exit and check potential obstacles.

Reflection (after image)

- Need to ensure proper alignment and navigation to exit buildings.
- Current focus is on successfully leaving the lab to progress the storyline.
- Stuck at the exit requires paying attention to visual cues for proper movement.

Action (after image)

UP, DOWN, DOWN

F PRACTICE GAMES

In this section, we expand on our three practice games defined in §4 and Table 4. We also verify that each game is solvable by humans playing through the VideoGameBench interface.

F.1 LOCATION CLICKING GAME

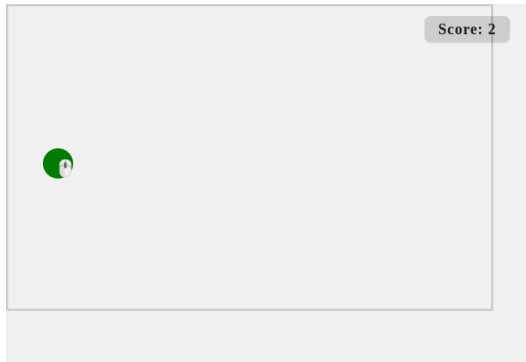


Figure 6: An example screen of the *Location Clicking Game*. A VG-Agent using a VLM is tasked with clicking 10 green circles, one at a time, in under 250 actions.

The most basic action in any DOS game is to click a position on the screen. The *Location Clicking Game* is a simple task where an agent must click a green circle with radius 40px that randomly generates inside a 640px by 400px region on the browser (this replicates the DOS game setting on VideoGameBench that runs inside a 640px by 400px window). Each time the agent clicks the circle, it respawns in a new location, and the agent must click the circle 10 times in under 250 actions. While some VLMs such as Claude Sonnet 3.7 and Gemini 2.5 Pro do not struggle with this task, many frontier VLMs require multiple correction steps to click the circle, and cannot complete the game in under 250 actions.

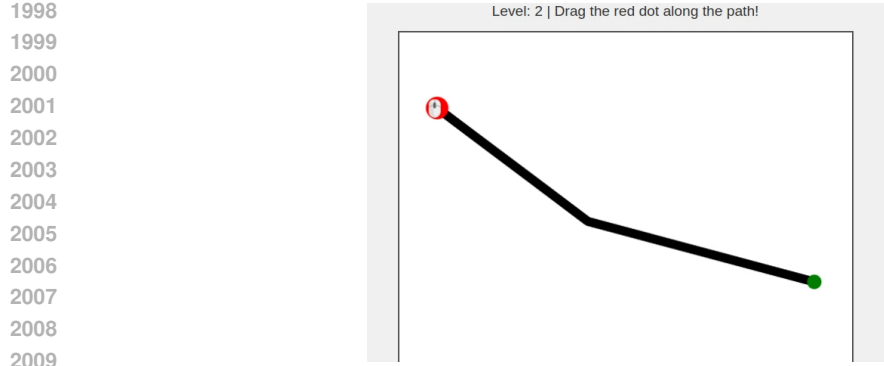


Figure 7: An example screen of the *Mouse Dragging Game*. A VG-Agent using a VLM is tasked with dragging a red circle to a target green circle while staying on the black line.

F.2 DRAGGING GAME

The VideoGameBench mouse and keyboard interface currently supports dragging the mouse from a position (x_0, y_0) on the screen to another position (x_1, y_1) in a straight line. Many games in VideoGameBench such as *The Incredible Machine* and *Age of Empires* require or allow dragging the mouse to play the game. On the *Mouse Dragging Game*, we devised 10 simple challenges for an agent to drag in a certain pattern. We find that all VLMs struggle on this challenge, with only Claude Sonnet 3.7 even completing the first level, which is just to drag in a straight horizontal line.

F.3 2D NAVIGATION GAME

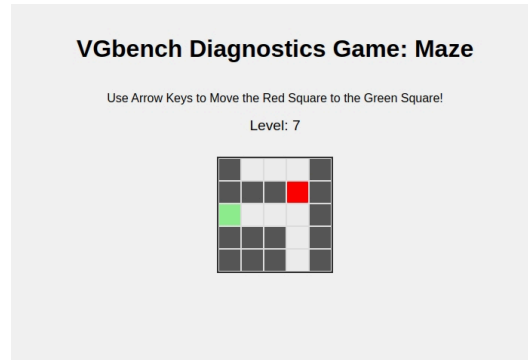


Figure 8: An example level in the *2D Navigation Game*. The task is to use the arrow keys (mouse + keyboard interface on VideoGameBench) to move a red square to the green square in a small grid-world maze.

In many DOS and Game Boy games in VideoGameBench, the agent must move a character in a 2D (often grid-world, e.g. *Pokemon Crystal*) environment. While this task is often easy for specialized agents, for VLMs it is not obvious that this task is easily solved. We generate 10 pre-defined mazes where the agent must move a red square to the green square in a small maze-like environment using the arrow keys. There are movable tiles (light gray) and immovable tiles (dark gray) that the agent must navigate. Although each maze can be solved in under 10 steps, we find that no agent can complete the entire game in under 250 actions, i.e. 25 steps per maze.