Coding with AI: from zero to hero in 2 hours

2026 SDSU Data Science Camp By Dr. Xijin Ge LinkedIn

Activity 1. Get Ready

1. Go to ChatGPT.com. Login is highly recommended.
2. Install Visual Studio Code (VS Code), a powerful and popular code editor.
3. Click on Windows Start button, then search for VS Code to start it.

Activity 2. Learn HTML, CSS, JavaScript

Prep:

- Create a folder called **learn** in the **Documents** folder.
- Start VS Code and open that folder. **File → Open Folder**.
- Create a new file in VS Code called hello1.html. **File → New File**

	Prompts	Note	File names
1	Act as a computer science professor. I am learning HTML. Give me the Hello World example.	Copy the code. Paste. Save. Double click on it from File Explorer.	**hello1.html**
2	Add a short paragraph about the history of HTML with a hyper link.		**hello2 links.html**
3	Help me understand CSS by changing the font and background color.		**hello3 CSS.html**
4	Show me how JavaScript works by adding a button to this page, which shows a message when clicked.		**hello4 CSS JS.html**
5	Customize as you like. (color, font, message …)		**hello5.html**

Note:

- Start small. Progressively improve.
- It is *essential* to name files and folders properly.
- Coding with ChatGPT involves trial-and-error by generating lots of versions.
- Remember where you files are.

Activity 3. To-do list web app

Prep:

- Create a new file in VS Code called **Todo v1.html** in the learn folder. **File → New File**

	Prompts	Note	File name
1	Write code for a to-do list app that runs in a web browser. Put everything in one file.		Todo v1.html
2	Write code for a to-do list that runs in a web browser. Tasks can be entered into a box. Once I hit enter, the task appears below the box, along with a checkbox. Put everything in one file.	**Revise** the first prompt. Do NOT continue. Mouse over the first prompt. Click on the pencil icon. 	Todo v2.html
3	Improve this app: - Make the title to "Zack's todo list" - Once a task is checked, change the font background to green. - Add an icon so that I can delete a task. Give me the entire code.	Follow up prompt. Notice how we structure the requests.	Todo v3 color.html

4	Once I complete a task, give me some positive feedback, such as sound and confetti. Plan first.	Plan with AI is essential for complex tasks.	
5	Implement. Give me the entire code.	Implement the plan	Todo v4 confetti.html
6	Customize by yourself: Add local time, timer, …		

Note:

- Plan and discuss with AI first before coding. This way you know what you are getting.
- One small step at a time. Don't eat a cow in one meal.

Activity 4. Shooter game.

	Prompts		
1	Write code for a simple shooting game that runs in the browser. Put everything in one file.	Resubmit a few times. See the difference in response.	shooter1.html
2	Write code for a shooting game that run in the browser. Put everything in one file. This game will feature a player at the bottom of a 400x400 playing area. The player can be moved left or right using the arrow keys. When the space key is pressed the player shoots a projectile. Targets appear randomly on top of the screen and move down. Targets disappear after being hit by projectiles.	Revise the first prompt. Resubmit to get a better working version.	shooter2.html
3	Change background color to dark grey.		shooter3.html
4	Instead of using the keyboard, the player moves horizontally using the mouse and shoots when I left click.		shooter4.html
5	Add a score counter.		shooter5.html
6	If the player is hit by the targets, the game is over. Users can restart.		shooter6.html

Learning outcomes:

- Resubmit prompts to get different code.
- Revise previous prompts.
- Refine iteratively

Activity 5. Snake game

Learning outcomes:

- When stuck, start a fresh conversation. Copy the existing code over.

Change the background to red in this app. My code is: XXXXXXX (paste existing code).

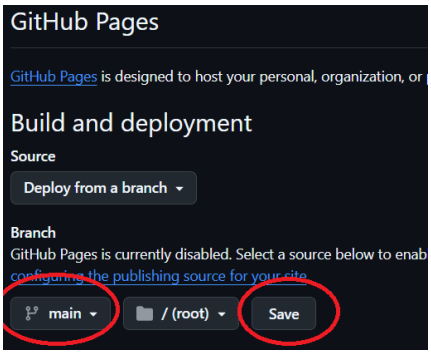
- Try other LLMs such as Google Gemini

	Prompts	Note
1	Write code for the snake game as one file that runs in a browser.	Resubmit a few times. See the difference in response.
2	The snake is just one square. Make it longer.	Revise the first prompt. Resubmit to get a better working version.
3	The snake went out of the boundary.	
4	Can we add a score reporting on the food count?	
5	Change the background to dark grey in this app. My code is: XXXXXX	Start a new chat!!!!! Include existing code in prompt.

Other games to try:

Game Title	Game Type	Prompt Example
Rock Paper Scissors	Logic game (2D buttons)	Write a Rock Paper Scissors game using HTML, CSS, and JavaScript.
Guess the Number	Text-based puzzle	Create a number guessing game in JavaScript where the computer thinks of a number between 1 and 100.
Tic Tac Toe	2D board game	Generate the code for a Tic Tac Toe game in HTML/JS that two players can play by clicking cells.
Hangman	Text-based puzzle	Build a Hangman game in HTML, CSS, and JavaScript. Include a list of words and display blanks for letters.
Quiz Game	Trivia/quiz	I want to create a simple quiz game in HTML/JS with 5 multiple-choice questions about [topic].
Pong	2D arcade game	Create a simple Pong game using HTML5 canvas and JavaScript.
Snake	2D arcade game	Generate a simple Snake game in JavaScript using a 2D grid.
Breakout (Brick Breaker)	2D arcade game	Create a Breakout-style game with HTML canvas and JavaScript.
Flappy Bird	Side-scrolling action	Build a Flappy Bird game in JavaScript with canvas, gravity, and pipes.
Space Invaders	2D arcade shooter	Create a simple Space Invaders game using HTML5 canvas and JavaScript.
Memory Card Match	Card matching puzzle	Create a memory card matching game in HTML/CSS/JavaScript with 8 pairs of cards.
Text Adventure	Text-based adventure	Make a browser-based choose-your-own-adventure game using JavaScript.
Whack-a-Mole	Clicker/reflex game	Write code for a Whack-a-Mole game in JavaScript with scoring.
Typing Speed Test	Skill game	Build a typing speed test in JavaScript with timer and error count.
Maze Runner	Maze puzzle	Create a maze game in HTML and JavaScript with keyboard controls.

Activity 6: Publish apps on GitHub.



Learning outcomes: Publishing games or websites on GitHub.

1. Create an account on GitHub.com
2. Create a repository called **shooter**.
3. From your computer, create a copy of the shooter6.html and rename it to **index.html**. Web pages normally default to this for landing pages.
4. Upload file. From the shooter repository on GitHub, click **Add Files** and upload the index.html.
5. Publish the game. Go the homepage of the new repository, and then click on **Pages** under **Settings**. Switch to the **Main** branches and click on **Save**.
6. Refresh this page after a few minutes. Your app is live at an URL like this https://gexijin.github.io/datamap/ Note the difference in the domain.

Activity 7: Create a data dashboard with GitHub Copilot

1. Create a folder in Documents called dashboard
2. Download the data file to this folder by right-clicking this link and choosing Save Link As (or Save Target As): heart attack data. It is also in our shared folder.
3. Start VS Code. **File → Open Folder** and choose the dashboard folder under Documents/.
4. Log in to your GitHub account by clicking on the "**Sign in**" button in the top center of VS Code.

5. Click the Chat icon to start GitHub Copilot. A chat window shows up on the right side.
6. Use this prompt:
   Create a dashboard for the heartatk4R.txt file. Use a button to upload the file so that I don't need a server.
7. Click on index.html from File Explorer to test the dashboard.
8. Improve it using this prompt:
   I want to highlight the difference between the sexes. Add a histogram to compare age distribution. And a bar plot to show death rate by sex.
9. Customize it on your own.

Now AI works on your computer. It wrote the code and saved it locally. Edit the code too. Your free GitHub account comes with free AI usage.

As you implement more features and chat more. Problems start to emerge: chats get too long and AI can get confused. We also need to keep track of our changes:

- Start new chat sessions (just upload you existing code file)
- Ask how to edit existing code instead of getting entire new code.
- Version control.

Activity 8: Create 3D games (Node.js)

1. Install Node.js: "Install Node.js" from GitHub Copilot. Yes, you can use AI agents to help install software. Or, you can Download the Windows Installer .msi file from https://nodejs.org/en/download
2. Start a Windows PowerShell terminal window from VS Code's main menu: **Terminal → New Terminal**
   On some machines, running commands are not allowed and the following commands give you errors. To solve that issue, right click on the Windows Start button on the lower left. Click **Terminal (Admin)** to start a PowerShell terminal. If it wants to upgrade, wait until it upgrates.
3. Bypass permission issue. In the VS Code Terminal or native Windows PowerShell Terminal:
   ```
   Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser
   ```
4. Start a new Vite template project. Lots of files are created in a new folder called **runner** under Documents/ folder.
   ```
   cd ~/Documents/
   npm create vite@latest runner -- --template vanilla-ts
   ```

   cd is short for change directory. **npm** is 'node package manager'. This also starts a web server at http://localhost:5173 by default. Once this URL shows up in the terminal, use **Ctrl + Click** to open it. You can also type http://localhost:5173 in a browser directly. Click on the counter to see if it works.
   If the page does not show, try to restart the server with: **npm run dev**

5. Start a new Terminal. From VS Code, Terminal → New Terminal. Alternatively, right click on the Windows Start button on the lower left. Click **Terminal (Admin)**. We keep the first terminal windows as a web server.
6. Install THREE.js for 3D games. From the new terminal:
   cd ~/Documents/runner
   npm install three @dimforge/rapier3d
7. Start VS Code. Click File → Open Folder → Open the **runner** folder under Documents.
8. Create a 3D endless runner game. From the GitHub Copilot chat on the left side:
   "Create a 3D endless runner game".
   Allow the agent to edit files. Once done, go back to the browser showing http://localhost:5173 to play it.
9. To build a new game, repeat steps 2-8. Rename the 'runner' folder to something else.

Activity 9: Version control using Git via AI agents

Now we have so many files to keep track of, and renaming them will break the logic. So we will need a software called Git to help us manage different versions of all files in a folder.

1. Download GitHub Desktop. Login using your GitHub account
2. From **File → Add Local Repository → Find the app directory**. You want to add the Documents/runner/ folder as a local repo. If you see error, click on "**Create a repository**"
3. Click **Publish your repository**. Uncheck **Keep this private** before submitting. All your files in the Documents/runner/ folder are uploaded to GitHub. You should be able to find this runner repository on GitHub.com after logging in.
4. From GitHub Copilot chat: "Start tracking changes".
5. Make a small change to the runner game using GitHub Copilot chat, such as changing a background color.
6. Then just say "Commit changes." Ask it to "Push to GitHub" to sync with the GitHub.com.
7. From the sidebar of VS Code click on the Source Control icon  , you should see the commits or restore points. You can also see the different versions from GitHub Desktop by clicking on **History**.

When coding with AI, it is essential to commit often so that we can revert when we or AI screw up a complex project.

Activity 10: Codex CLI, a powerful coding agent

1. Open a new terminal. From VS Code, Terminal → New Terminal. Or, Right-Click on Windows Start button, click **Terminal (Admin)**. You can also reuse the 2nd PowerShell Window.
2. Install OpenAI's Codex CLI with this command: **`npm install -g @openai/codex`**
3. Optional: Install the Codex extension to VS Code. From the sidebar on the left, click on **Extensions**. Or Ctrl + Shift + X. Search for **Codex** and install the one from OpenAI that has millions of downloads. Then From VS Code main menu, select View → Chat to open the chat window on the right side. Then click on the OpenAI icon to start Codex. Login using your own OpenAI account or using my API key below.
4. Optional: From Terminal (either in VS Code or directly in Windows PowerShell), navigate to your game folder.
   cd ~/Documents/runner
   `codex --yolo`
   We start Codex in yolo mode, granting all permissions to Codex AI agent. This is efficient but a bit risky.
5. Select **API key** to log in. Then paste this key. To get your own API key, go to https://platform.openai.com
   sk-proj-sc8jorKFMMWM_... -olRXJhHV4OMA

6. Once logged in. Switch model to gpt-5.6-terra typing the **/model** command and choose **"2. gpt-5.6-terra".** This is just to avoid the most expensive model.
7. Just ask it to build games for you. For example, with a prompt like this:
   "Improve the 3D runner game by making the background more realistic.".
8.

Publish apps on GitHub

1. Start GitHub Desktop. Make sure you are logged in using your GitHub account
2. Add Repo from local file: Find the app directory.
   If you see error, click on "Create a repository"
3. Click "Publish your repository", Uncheck "**Keep this private**"
4. In Codex started from the project folder: "Commit changes." "Push to GitHub"
5. In Codex:
   "Configure this project for deployment via GitHub actions. Commit and push."
6. On GitHub, click on this repository, then **Settings → Pages → Source → GitHub Actions**.
7. On GitHub: Actions → Click on the error →Re-run all jobs
8. Wait 1 minute: **Settings → Pages**. Click on the link

Tips:

- Keep chat sessions focused and short. Use the **/clear** command to start new session.
- One chat session per feature.
- "Create a AGENTS.md with a brief overview of this project.
- Commit and push often!

Other 3D games to try

Genres that broaden appeal without narrowing anyone's:

- **Decoration / room designer** — place furniture, change materials and colors, save the layout. Sounds slight, but it's real 3D work: raycasting, object placement, transform manipulation, serialization. And it produces something students *want* to screenshot.
- **Cozy management sim** — a café, a plant shop, an aquarium. Timers, resources, upgrades, growth curves. The systems thinking is genuinely deep and it looks nothing like a shooter.
- **Rhythm game** — consistently one of the most gender-balanced genres there is, and the audio-sync problem is a great technical challenge.
- **Photography / exploration game** — wander an environment, find and frame specific things, get scored on composition. No combat, all camera work and level design.
- **Puzzle games** — portal-ish, block-pushing, light-reflection, gravity-flipping. Pure logic, huge design space, very shippable.
- **Co-op or asymmetric two-player** — one on keyboard, one on mouse, both needed. Same couch, no competition.
- **Pet / creature raiser** — customize a creature, care for it, watch it change. Enormously appealing and mechanically simple.
- **Dress-up / avatar customizer as a *component*** — bolt it onto any other game. Cheap to build, disproportionate engagement.

Target range / shooting gallery. Pointer-lock camera, raycast on click, targets that pop up and score by accuracy and time. Technically the simplest possible shooter: no enemy AI, no health, no navigation. Students learn raycasting, mouse look, and hit feedback in a day.
Snowball or paintball arena. Projectile physics instead of hitscan — you throw something with an arc and lead your target. Genuinely more fun than hitscan for beginners because misses are readable, and it gives Rapier a real job.
Wave survival vs. blobs/robots/slimes. The classic "horde" loop. Enemies spawn at the edges, walk at you, you knock them out, waves get bigger. The AI is `moveToward(player)` — about four lines — and it feels like a real game immediately. This is probably the highest fun-per-line-of-code of anything on this list.
Asteroids in 3D. Free-flight in space, shoot rocks, they split. No ground, no gravity, no navmesh — a lot of the hard parts of 3D just vanish.
Tower defense. Place turrets on a grid, enemies follow a fixed path, turrets auto-target and fire. Shooting without a shooter, and the strategy layer means students spend their time on balance math rather than aim mechanics.
Beyond shooters, the ones that reliably ship in a week:

- **Kart racer with ghost times** — race your own replay instead of building netcode
- **Marble / mini-golf physics** — Rapier does the work, students design levels, which is the actually creative part
- **Parkour time trial** — moving platforms, checkpoints, a clock; a direct extension of the runner they already have
- **Rhythm game** — notes on a track, hit on beat; audio sync is a great hard-but-bounded challenge
- **Flappy-bird-style one-button** — trivially simple, and a great "ship something in 90 minutes" warm-up on day one
- **Physics sandbox / knockdown** — stack things, launch a projectile, score the destruction. Pure Rapier showcase.

Topics covered:
Editor: VS Code
Web development: html, CSS, JavaScript, Node.js, Three.js
Command line: Windows PowerShell, Terminal from VS Coe
Code generation: ChatGPT.com, GitHub Copilot, Codex CLI
Version control: Git (via coding agents), GitHub.com, GitHub Desktop
Publishing: GitHub Pages, GitHub Actions

Updated 8/1/2026 at Hopkins, MN