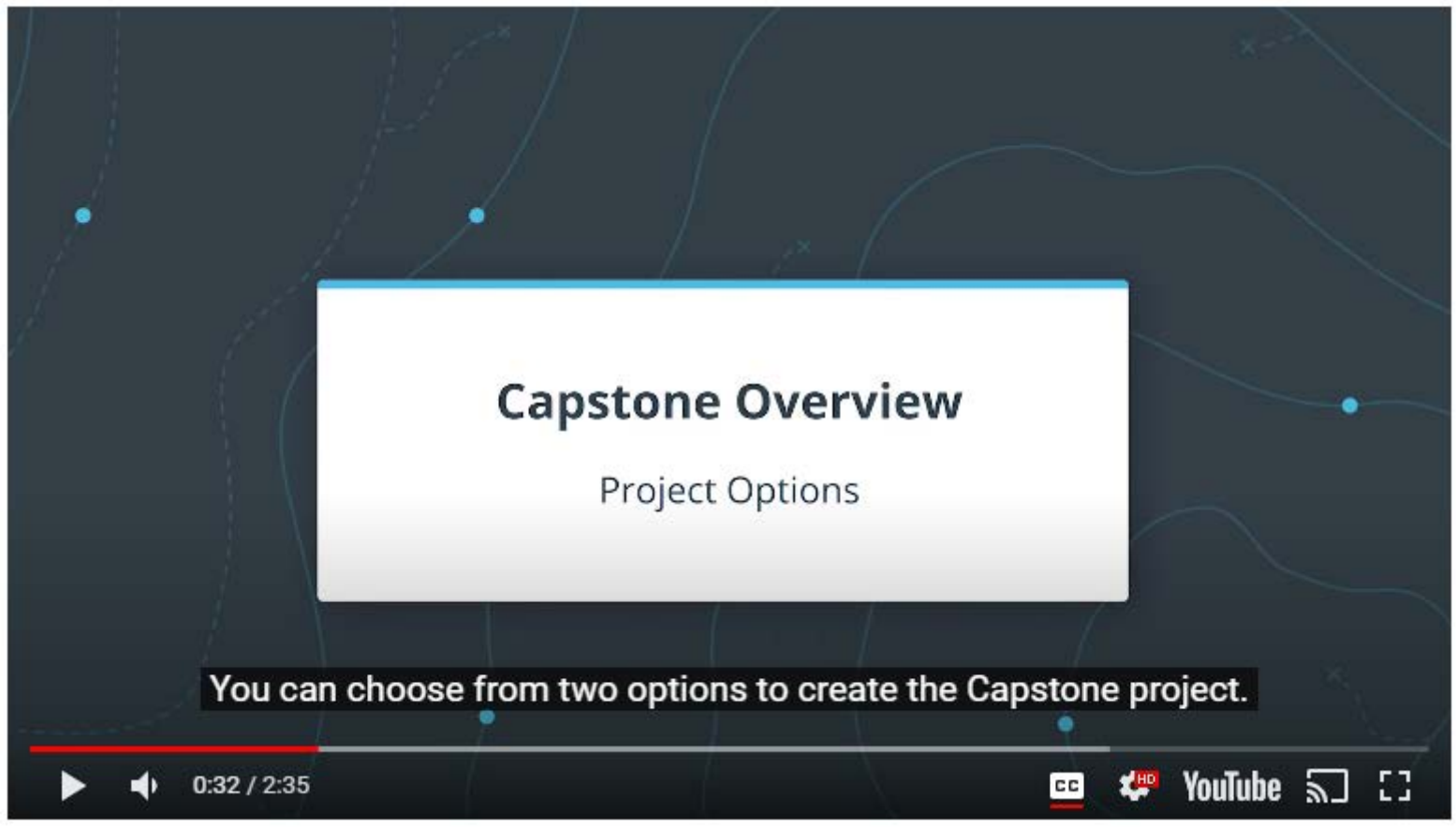
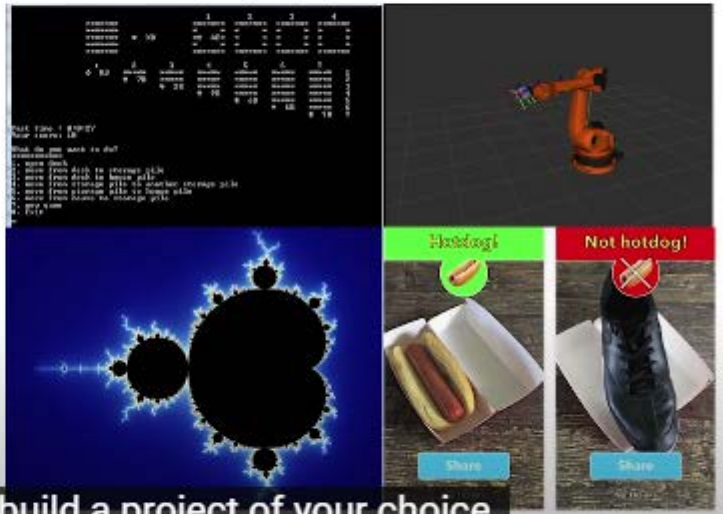

Capstone Overview
Project Options
You can choose from two options to create the Capstone project.
0:32 / 2:35
YouTube

Option 1: Build the Application of Your Choice

Examples:

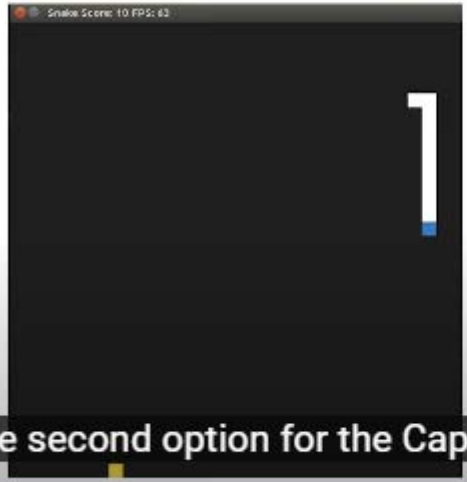
- Chatbot
- Neural network deployment
- Text-based solitaire game
- Microcontroller for an embedded system
- ROS Node simulated in Gazebo
- Networking protocol
- Mandelbrot or Fractal creator
- Hot dog detector using OpenCV



The first option is to build a project of your choice.

Option 2: Code a C++ Video Game

Add a feature to Snake:



Build from scratch:

- Asteroids
- Pacman
- Space Invaders
- Pong

The second option for the Capstone project is to add features to a video game.

Overview

The Capstone Project gives you a chance to integrate what you've learned throughout this program. This project will become an important part of your portfolio to share with current and future colleagues and employers.

In this project, you will build a C++ application, following the principles you have learned throughout this Nanodegree Program. This project will demonstrate that you can independently create applications using a wide range of C++ features.

You can choose from two possible options for this capstone project:

Capstone Option 1

The first option is to build the project of your choice. This is the most flexible option, as this will allow you to focus on the type of application that interests you most. There are a wide variety of applications you can choose from, but here are some suggestions for you to think about:

- Chatbot
- Neural network deployment
- Solitaire
- Microcontroller for an embedded system
- ROS Node simulated in Gazebo
- Networking protocol
- Mandelbrot or Fractal creator
- Hot dog detector using OpenCV
- Video game

There are a wide variety of existing projects and tutorials online that can help you get started, and we'll provide some links below to help. We'll also provide a code repository with a CMakeLists.txt and a "Hello World" application that you can build from. You can use this repo to get started if you wish, but you aren't required to use it.

Capstone Option 2

The second option for the capstone is to add features to a video game. We'll provide you with a simple 2D Snake game that was inspired by this Stack Overflow post on beginning game development with the SDL game programming library. The game code provides a solid foundation for you to extend with additional features, or you are welcome to use the code as a framework to build you own game from scratch. If you decide to build your own game, cloning a classic 2D game is great way to get started. Possible ideas include:

- Asteroids
- Pacman
- Space Invaders
- Pong

The game you create doesn't need to be a fully-featured clone of the original, but there should be enough of a game for a user to play.

Both project options will require a solid effort from you to complete, but I am confident that you will be successful, and we can't wait to see what you create!

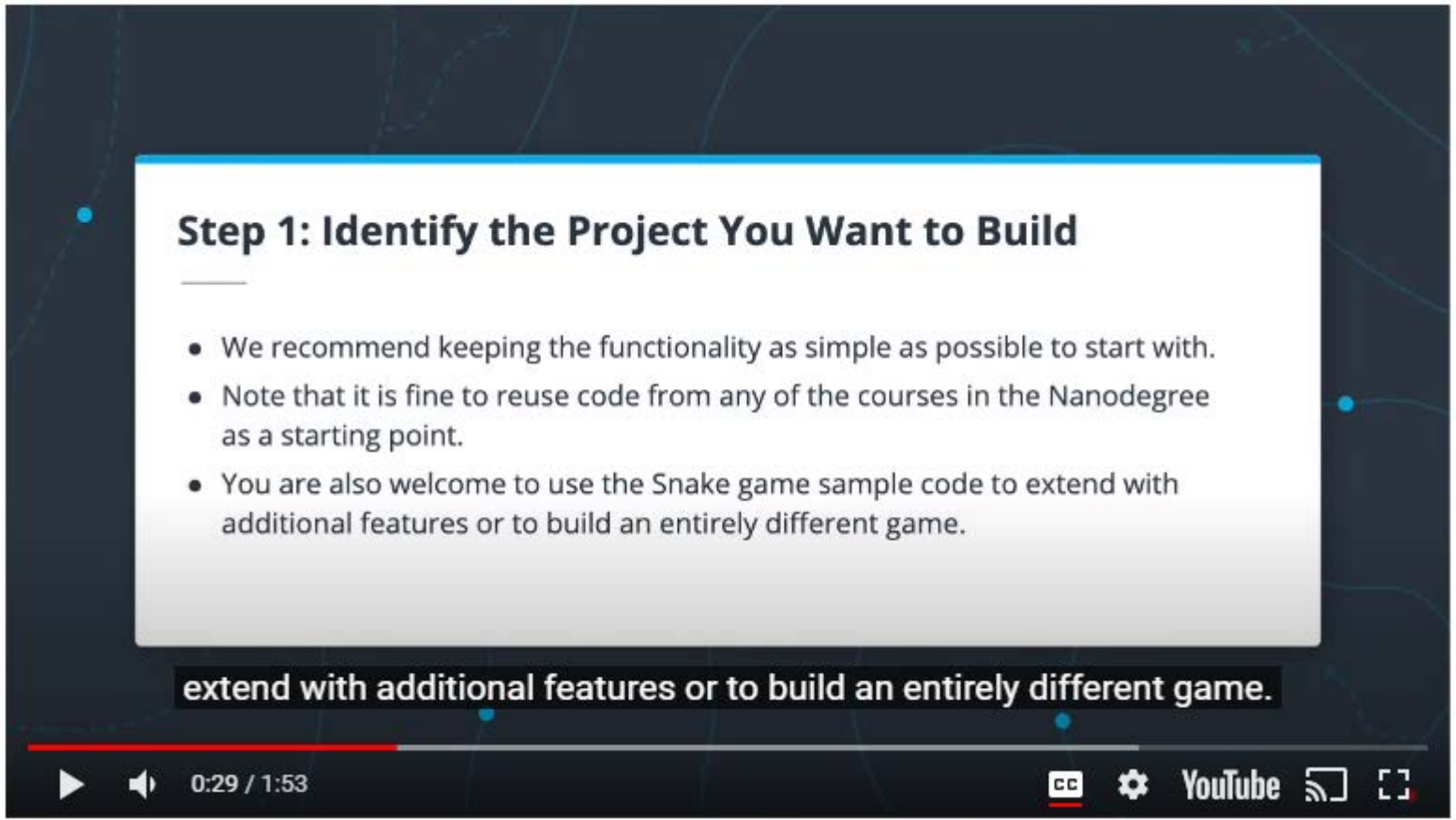
Resources

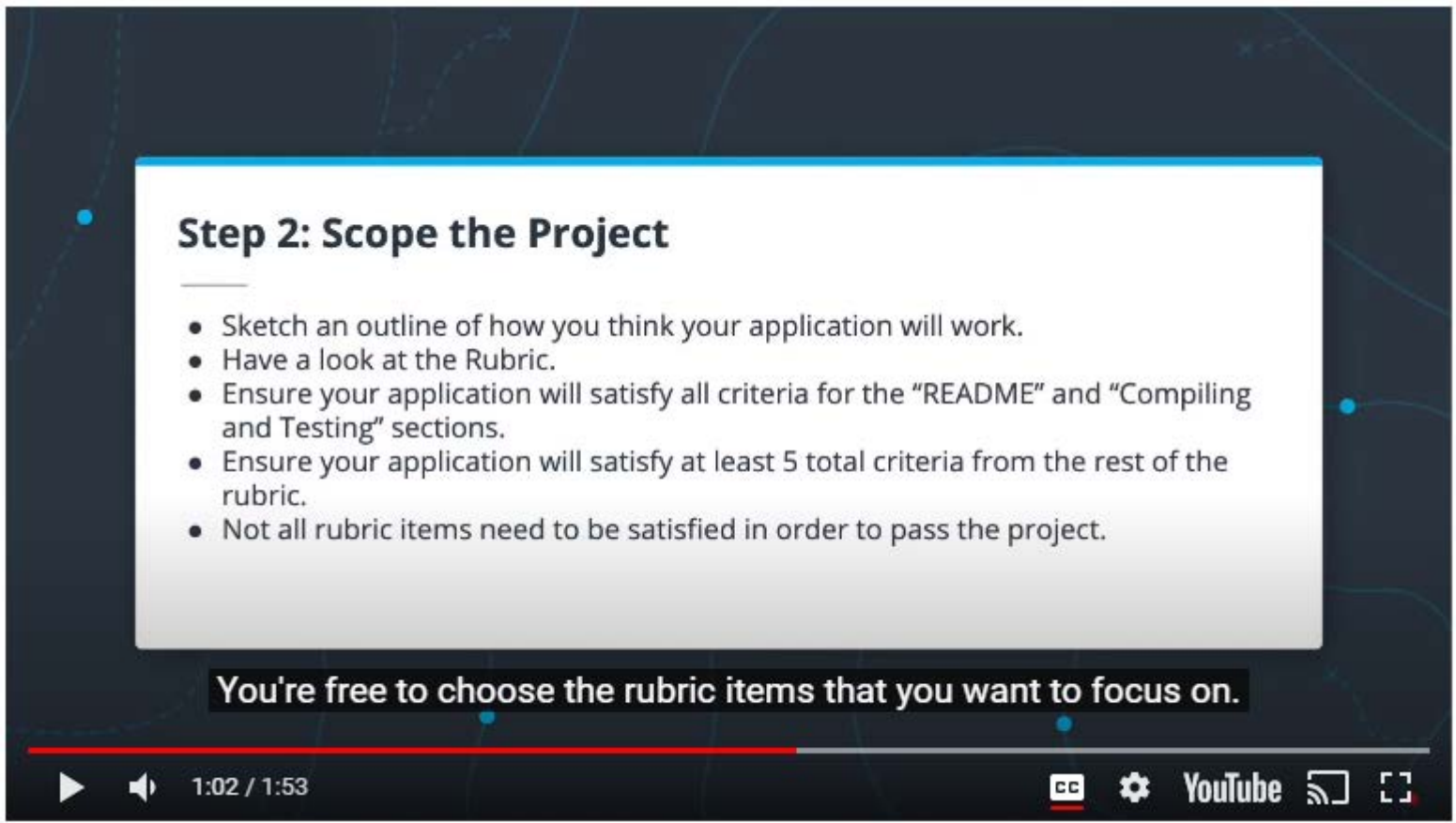
Code Repo and Rubric

- The 2D Snake game Github repository is here.
- The empty starter Github repositry is here.
- The project rubric is here. Note that not every rubric item needs to be satisfied to pass the project. Detailed instructions are provided in the next classroom concept.

Selected Project Resources

- Tensorflow C++ Api
- ROS (Robot Operating System) Tutorials. Some of the tutorials here use C++ and some are in Python.
- Mandelbrot set tutorials here and here
- Game programming resources will be provided in the upcoming concepts.

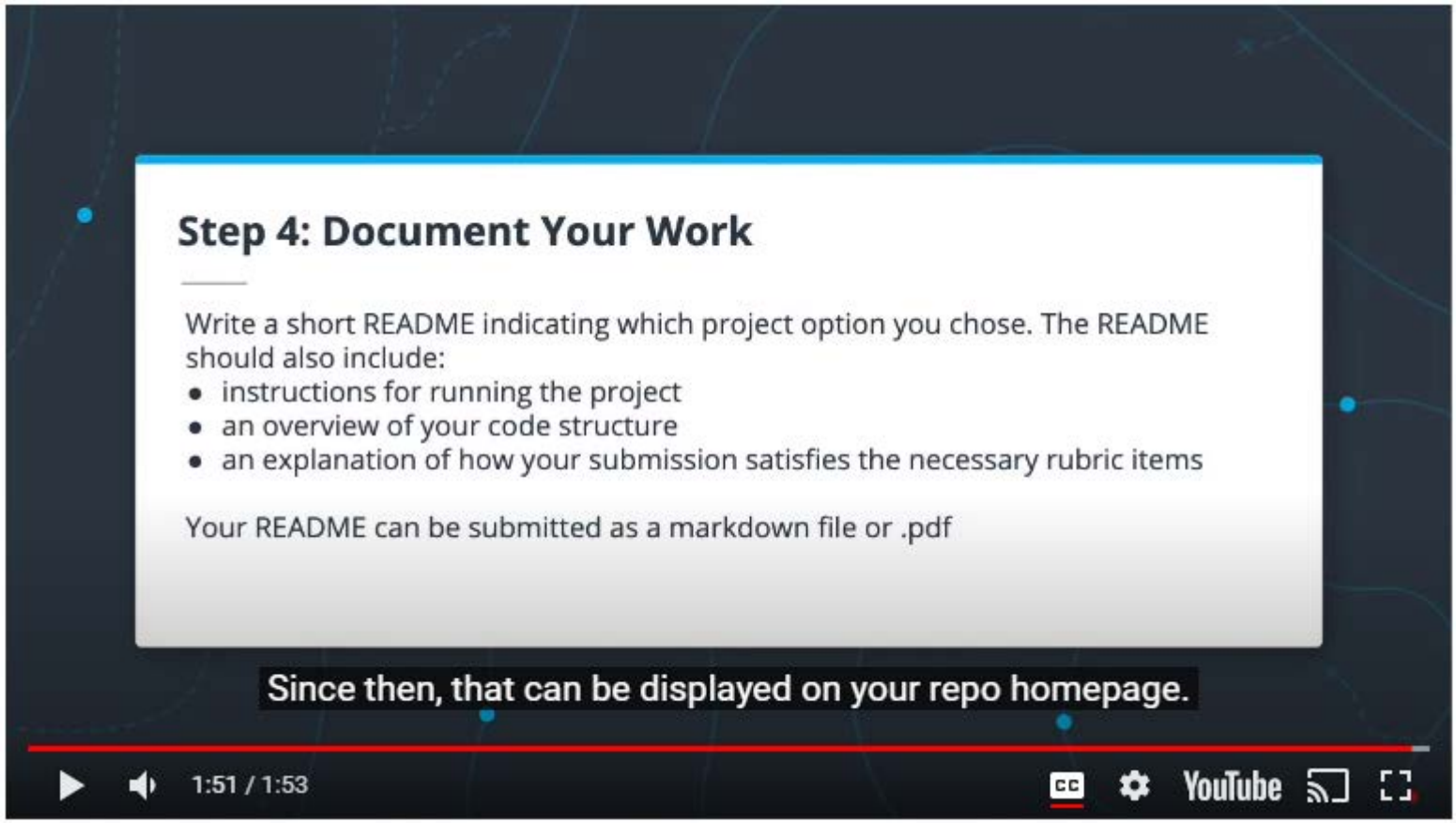

Step 1: Identify the Project You Want to Build
We recommend keeping the functionality as simple as possible to start with.
Note that it is fine to reuse code from any of the courses in the Nanodegree as a starting point.
You are also welcome to use the Snake game sample code to extend with additional features or to build an entirely different game.
extend with additional features or to build an entirely different game.
0:29 / 1:53
YouTube


Step 2: Scope the Project
Sketch an outline of how you think your application will work.
Have a look at the Rubric.
Ensure your application will satisfy all criteria for the “README” and “Compiling and Testing” sections.
Ensure your application will satisfy at least 5 total criteria from the rest of the rubric.
Not all rubric items need to be satisfied in order to pass the project.
You're free to choose the rubric items that you want to focus on.
1:02 / 1:53
YouTube

Step 3: Build Your Application

Hopefully, this part will be the most fun.

1:13 / 1:53


Step 4: Document Your Work
Write a short README indicating which project option you chose. The README should also include:
• instructions for running the project
• an overview of your code structure
• an explanation of how your submission satisfies the necessary rubric items
Your README can be submitted as a markdown file or .pdf
Since then, that can be displayed on your repo homepage.
1:51 / 1:53
YouTube

Recap of Steps

Step 1: Propose a Project

Identify the application you want to build. We recommend keeping the functionality as simple as possible to start with. Note that it is fine to reuse code from any of the courses in the Nanodegree as a starting point. You are also welcome to use the Snake game sample code to extend with additional features or to build an entirely different game. If you prefer to start from scratch, you can find a starter repo here.

Step 2: Scope the Project

Review the rubric. Ensure that the application you build **will satisfy all criteria for the "README" and "Compiling and Testing" sections**, and that the application **will satisfy at least 5 total criteria from the rest of the rubric. Not all rubric items need to be satisfied in order to pass the project**. Scope the features of your project to meet the rubric criteria you want to target.

This flexibility will allow you to focus on the aspects of your application that interest you most.

Step 3: Build your application

Build your own application, or extend the Snake game with new features.

Step 4: Document Your Work

Write a short README indicating which project option you chose. The README should also include:

- instructions for running the project
- an overview of your code structure
- an explanation of how your submission satisfies the necessary rubric