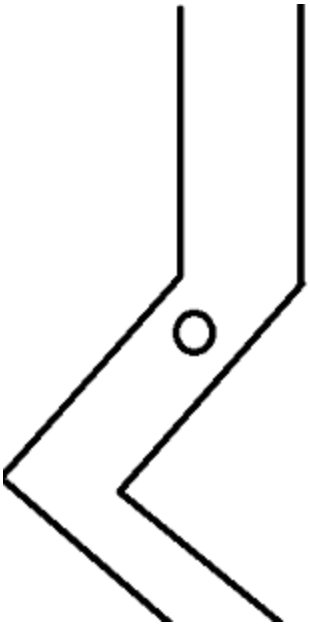
# ECE540: Final Project Proposal

*Tunnel Vision*

Erik Rhodes
Bhavana Dhulipala
Rohan Deshpande
Nikhil Patil

## Project Description

The basic objective of *Tunnel Vision* is to navigate through a tunnel that zig zags left and right. The walls will continuously get narrower until the player hits one of them. The score will be displayed on the screen, and there will be options for different difficulty levels (with the harder levels employing a faster speed). The player will use the left and right buttons on the Nexys 3 board as controls. The tunnel and icon will be displayed on a VGA monitor so that everyone can watch as the player inevitably screws up.



## Design Approach

The Picoblaze core will interact with pushbuttons (our inputs) to generate and handle the interrupts needed to implement a smooth game flow. The location of the player and the tunnel walls will be sent to the video display module through an interface module. While the position of the player will appear to be moving down through the tunnel, it will remain at the same height on the screen. The tunnel walls will be moved upwards at a specific speed. When the user moves sideways, it will travel across the display monitor (our output) one x-coordinate at a time in the appropriate direction.

Collision detection will simply check the x-coordinate of the player's position and compares it with the tunnel walls. If the value of the player's x-coordinate is on or outside the wall's x-coordinate, the game and score will stop. The reset button can be used to restart the game.

The walls can be vertical or can be slanted at 45 degree angles left or right. A Linear Feedback Shift Register can be implemented to get pseudo-randomized 0's and 1's to control the slant of the tunnel, and the width of the tunnel would be a function of time, getting harder as we went. The randomization might need to be weighted so that we don't get a bunch of very small changes in direction. Longer, smoother tunnel changes might be more ideal.

The score will be calculated based off how far the player gets. We will keep track of using a counter every time the next location is generated for the wall. If we implement different difficulty levels, however, more logic will be needed to make the scoring fair. Since an easier (slower) would allow the player to get further through the tunnel, we will include a multiplier proportional to the speed of the selected game.

The icon will be generated based on its location and orientation. Initially, the icon designed will be symmetrical . Given there is sufficient time after the game is fully functional, the icon can be changed to be more visually interesting. By default, the icon will be facing down. If the left or right button is "currently" being pressed, the icon will rotate left 45 degrees in the appropriate direction. If it is not being pressed, it will resume facing downwards.

The display monitor is implemented with a VGA connection using a video controller module similar to the one used in the RojoBot project, and as it is quite similar, will not be explained in more detail here. Likewise, there are more complex implementations and stretch goals we are considering that are shown further down.

The program will be considered a success when it is able to:

- Generate a random (but weighted) tunnel that moves left and right
- Shift the map at a certain "speed" depending on the level of difficulty
- Decrease the size between the walls during gameplay
- Generate the icon and its movement logic
- Control the icon's movement by interfacing with the manual pushbutton controls
- Keep track of the game time and score
- Detect when the player collides with a wall to stop gameplay

## Timing Constraints

If generating an adequate random course is too difficult, we have a few fallback options:

- Pre-program several tunnel courses to cycle through.
- The task of programming different difficulty levels can be dropped
- Icon image can remain simple

In addition to these options, we have included many possible improvements to integrate. The following implementations can be dropped if the tasks prove too difficult:

### Stretch Goals

- **Extra game additions that adds more options/rules/visual appeal**
    - Display of previous high scores
    - Opening splashscreen
    - Powerups/Bonuses (slowing down time, extra lives, etc)
- **CMUCam peripheral used to add video processing elements**
    - Element of game is controlled by visual surroundings
    - Player's icon can be controlled by visual object (gloves/ball of a certain color?)
    - Security enhancements made from visual verification
- **Nintendo Gamepad Controller**
    - Control the motion of the player
    - Needs NES gamepad controller adapter to USB
    - May prove to need too much extraneous work/hardware
- **Integrate DAC peripheral for audio enhancements**
    - Audio feedback given for positive/negative events
    - Soundtrack plays for the duration of the game
    - Intro Music
    - PMOD DAC component needed

## Milestones/Deliverables

### Week of 2/23

- Project Proposal submitted on D2L (by 10PM 2/27) ✓
- High level planning done
- Game logic flowchart created
- Block diagram/schematic created
- All extra hardware components ordered
- Github repository created ✓

### Week of 3/2

- Simple Icon image programmed
- Tunnel walls generated
- Randomized direction of walls generated
    - Look into using LFSR
    - May need to be weighted to ensure smoother gameplay
- Program player's control of icon

### Week of 3/9

- Progress report presentation (3/11 in class)
    - Discuss progress and challenges with slides
- Collision detection employed

- Game logic implemented
    - Changing speed of game
    - Timing on when to shrink walls
    - Any bonus rules/options/visuals
- Visually appealing icon image created
    - Logic to rotate and display icon adjusted
    - Orientation of icon controlled by pushbuttons implemented in Picoblaze firmware
- Finish creating a working project, start on stretch goals
    - CMU camera implemented
    - Soundtrack added
    - Nintendo controller

## Week of 3/16

- Finish stretch goals previously decided on
- Ensure fully functioning project
- Prepare/give final presentation and demo (3/18 in class)
- Write up/submit project report and documented code on D2L (by 10PM on 3/20)