


# CS486C – Senior Capstone Design in Computer Science

## Project Description

<b>Project Title:</b> Computer vision for logging the results of polyhedral dice rolls	
	<b>Jason Robinson</b> Beautymark Design Studio jason@beautymarkdesignstudio.com (646) 256-9059  <b>Toby Dylan Hocking</b> Assistant Professor SICCS <a href="mailto:Toby.Hocking@nau.edu">Toby.Hocking@nau.edu</a> (928) 523-5209

### Project Overview:

Tabletop roleplaying games (TTRPGs) are currently the fastest-growing category within the burgeoning “hobby games” industry (up 18%, from \$55 to \$65 million, in 2018).

The iOS and Android app stores are filled with apps related to TTRPGs, but are largely not addressing one particular aspect of the hobby: using technology to facilitate face-to-face gameplay. Beautymark Design Studio is currently in the design and planning phase of an application suite intended to address this gap. Our vision is to create mobile software that either a) streamlines the experience of players by offloading complexity or arduous busy-work to the app or b) increases the immersion of gameplay without increasing complexity. In short: better stories, told more easily.

These apps are not games themselves, but *helper apps* to facilitate the play of other games. The goal is not to create a situation where a bunch of humans are sitting around a table staring at their phones; rather, for players to quickly glance at the app then return to their face-to-face gameplay. In this sense, they are more like utility or productivity apps than like games. A really good app would provide tools/automations of processes that appear in *many different* games, i.e., it could be used as a utility by players to streamline playing all such games.

One specific challenge in the mixed (digital+analog) space is the role of **dice**. As a game mechanic, any die roll can of course be suitably replaced by a random number generator. The challenge is cultural: *gamers love their dice*. If we want to create a hybrid environment that benefits from software, but still supports a physical space, gamers need the option to *physically roll their own personal dice* — and have the results accurately and immediately logged in the software. This has a side benefit related to storytelling: at the end of the session, an outline of key actions taken (anything which requires a die roll from players) has been automatically generated, which can be used for data analytics and narrative purposes.

Here is a sample scenario:

1. Lyric indicates to the Game Master that her character Rojan intends to pick the gerent’s pocket — and that she’s willing to apply some extra effort in order to succeed.
2. The GM, Alyx, replies that a roll of 14 on a 20-sided die (d20) will be required — but she can spend effort to add a d4 to the roll. The GM assigns a quick label to this roll: “pick gerent’s pocket”.
3. Lyric rolls 1d20 + 1d4, selecting these personal dice from her own collection.
4. A dedicated mobile phone camera at the table observes the area into which the dice were rolled, and uses ML to accurately reports their totals: 11 on the d20, and 2 on the d4, totaling 13.

5. Lyric has just missed her required result of 14 — her character gets caught with his hand in the gerent's pocket. The GM's app logs metadata, eg: "**Rojan attempts to pick gerent's pocket but fails with a 13/14, despite adding 1d4 effort.**" (*To be clear: network communication between apps, etc. is not part of the scope of this request — this is just an example of how it might be used in a final product.*)

Many apps exist to roll virtual dice. I have found Machine Learning (ML) models online for evaluating the results of die rolls — but almost always six-sided dice (with dots, not numbers). I have a good idea of how to approach many other aspects of the app, but the ML/computer vision aspects of this are not something I can hope to address myself. Here is what I am looking to produce:

- A standalone app that can monitor a video feed in realtime, watching for one or many polyhedral dice rolled into the frame, then recognizing and recording: a) Number of dice b) Dice type by number of sides (D2, D4, D6, D8, D10, D12, D20) c) Value of die rolls

This is challenging, for a few reasons. Although we may be able to provide general guidelines ("the app will work better if the die-rolling area is well-lit"), it ultimately will *not be adequate* to get this working only in a tightly-controlled lab environment. In use, players will have custom dice, variable backgrounds, different cameras, unpredictable lighting conditions, angles of view, etc. — and of course, dice don't always land facing toward the camera. So, a proof of concept solution that works in ideal situations is a good start, but a real solution needs to be *incredibly* robust in order to have any chance of being adopted.

The potential applications for this (as an ML model or software library, which could potentially be made available to other apps) are pretty exciting. Polyhedral dice are used as a core mechanic of *almost every* TTRPG out there.

### **Solution Details**

A new mobile application running on a modern Android operating system. Ideally, implementation would explore using one of the new cross-platform application frameworks (e.g. Ionic, React Native) that (supposedly) allow single-codebase deployment of an app to both Android and iOS. For this proof of concept, however, we will prioritize Android due to ease of development/deployment. Specific features and development goals are straightforward and include:

#### **Minimum features (Minimum viable product):**

- User can take a picture of rolled dice, and have the results interpreted.
- Resulting values are logged: total number of dice, of which shape, results, and optionally a label.
- Able to use ML to recognize the die rolls on black or white standard polyhedral dice (d4, d6, d8, d10, d12, d20) *\*\*in ideal lighting\*\**. Proves ML basically can do this.
- A basic interface for training the AI system, i.e., a way to point it at a folder of training images that have been human-read/classified and use them to train the AI system.

#### **A well-appointed app (something that might actually be used!):**

Improvements to any of the basic features above, as discovered in requirements acquisition. Could include:

- Monitors a phone camera's video, recognizing when players roll dice and delivering results once the dice come to rest.
- Better/more efficient AI training system, e.g., a way for humans everywhere to submit pics of rolls as well as human-read results that can be used to train the system.
- Ability to handle non-ideal lighting conditions and a wide range of (nonetheless legal) dice (colors, sizes, etc).
- More robust and friendly GUI that actually supports the game playing process, e.g.,
  - Can register new players (and their apps) into the system
  - Allows creation of new "games", with specific registered players attached to a particular game
  - helps log/record rolls, outcomes, and notes, as illustrated above.
  - (Client can supply wireframes for this GUI if desired.)

**Stretch goals:**

- Provides support for one or more participants to be remote, i.e., playing via a Skype link. In this case, rolls by all players are broadcast to all members along with outcomes and notes recorded.
- Network connectivity, allowing GM to request a roll, and the results to be returned from the camera device to the GM and player devices.
- Other cool features discovered as useful/fun but peripheral during requirements acquisition.

**More about the client**

Finally, a few notes about me as a sponsor — I'm a user experience designer, and can contribute as much as is helpful to the UI design: low-fidelity wireframes, high-fidelity comps, motion prototypes, etc.

I am *incredibly* excited about working with a team of skilled developers who care about the overlapping Venn of computer vision, machine learning, gaming, native mobile app development, and great storytelling.

**Knowledge, skills, and expertise required for this project:**

- Native mobile application development.
- Machine Learning, especially as applied to Computer Vision
- Software design, development, and usability
- Probabilities
- An understanding of game design or storytelling theory may be helpful for feature ideation

**Equipment Requirements:**

- A mobile development platform, e.g., XCode for iOS development.
- Machine Learning software for training and developing models (also freely available)
- Cameras and other equipment for image capture. Cell phone cameras or other conveniently available.
- Polyhedral dice. Sponsor will provide a set if needed, but team members are encouraged to get a set of their own!

**Software and other Deliverables:**

- The prototype mobile application, as described above, deployed and tested successfully with real data. Must include a complete and clear User Manual for configuring and operating the software.
- A strong as-built report detailing the design and implementation of the product in a complete, clear and professional manner. This document should provide a strong basis for future development of the product.
- Complete professionally-documented codebase, delivered both as a repository in GitHub, BitBucket, or some other version control repository; and as a physical archive on a USB drive.
- A machine learning model, exported in (or converted to) Apple's **Core ML** format.
- A training set of labeled images or videos that can be used to train future ML models.