Piranha Plants as Charade: Exploring Melody-Guided, Algorithmic Music Generation

Max Huang*, Emily Yu*

*Equal contribution

Summary

The premise of Piranha Plants as Charade is to transform a melody into a full-fledged song in the style of "Piranha Plants on Parade" from the video game "Super Mario Bros. Wonder". Given an input melody in the form of a digital signal, we developed a music generation algorithm that transforms the input with the following process: 1) it extracts the melodic information from the input; 2) it generates a chord progression that fits under both the extracted melody and the appropriate stylistic conventions; 3) it generates a musical arrangement based on the melody and chord progression; and 4) it exports the generated arrangement as a WAV file. For clean inputs within the targeted scope, our algorithm can often produce convincing results, but for a majority of the cases, the output is subpar. Step 1 (melody extraction) is most likely the largest source of failure; the pitch detection is generally correct, however, the onset detection often produces false positives. Despite the current shortcomings, we believe that the core ideas behind "Piranha Plants as Charade" can be extended to adequately meet our premise. We plan on fine-tuning our algorithm and further iterating on our program to produce better and more consistent results.

1 Introduction

1.1 Motivation

The fields of mathematics and music are heavily intertwined. In the late 16th century, many Western European music theorists believed that they had developed *ars perfecta*: a set of rules for which, if followed, guaranteed that music be "free of reprehensible elements, purged of every error and polished, and [the] harmonies will be good and pleasant" [1]. Although the concept of perfect music is a footnote in the modern musical landscape, we are intrigued by their aspirations to model musical correctness with rule-based approaches. We wanted to similarly explore the relationship between music and mathematics by generating music using rule-based algorithms that humans follow — processes that mirror the human composer. Our goal was to create a proof-of-concept computer program that transforms a melody into a full-fledged song in the style of "Piranha Plants on Parade"[1] from the video game *Super Mario Bros. Wonder*. We call this project Piranha Plants as Charade as a reference to how we imitate the original song.

Why "Piranha Plants on Parade"? We believe that this is the ideal song to emulate for our proof-of-concept. It is based on a relatively simple harmonic framework, which allows us to focus on developing the algorithm's high-level concepts rather than on hard-coding harmonic rules; yet the transitions between harmonic states are distinct enough to be recognizable. Similarly, the song's accompaniment style is repetitive enough that it can be approximated with only a few rules. "Piranha Plants on Parade" also distinctively features gibberish lyrics, which allows us to explore voice generation without worrying about conforming to an existing language. Lastly, the song originates from a video game, a medium with a long history of using synthetic audio and music, thus we believe it to be a thematically appropriate subject for computer-generated music.

1.2 Minimum Viable Product (MVP)

We worked on Piranha Plants as Charade for six weeks, borrowing ideas from several areas of computer science and music, including but not limited to: digital signal processing (DSP), hidden Markov models (HMM), Western music theory, and jazz performance. Due to time and budget limitations and our lack of expertise across our explored domains, we believed it to be unreasonable for us to perfectly emulate the style of "Piranha Plants on Parade". Thus, our goal was to create an MVP, which we defined as follows:

1. The MVP should support melody inputs with the following properties:
   (a) Has a time signature of 4/4.
   (b) Is in the key of C major.
   (c) Has a tempo of 110 beats per minute.
   (d) Starts on a note (i.e. not a rest).
2. For any supported input, the MVP should output a song with the following parts:
   (a) The input melody and a harmony line, both sung in a 'Piranha Plant' style.

[1] `https://www.youtube.com/watch?v=3EkzTUPoWMU`

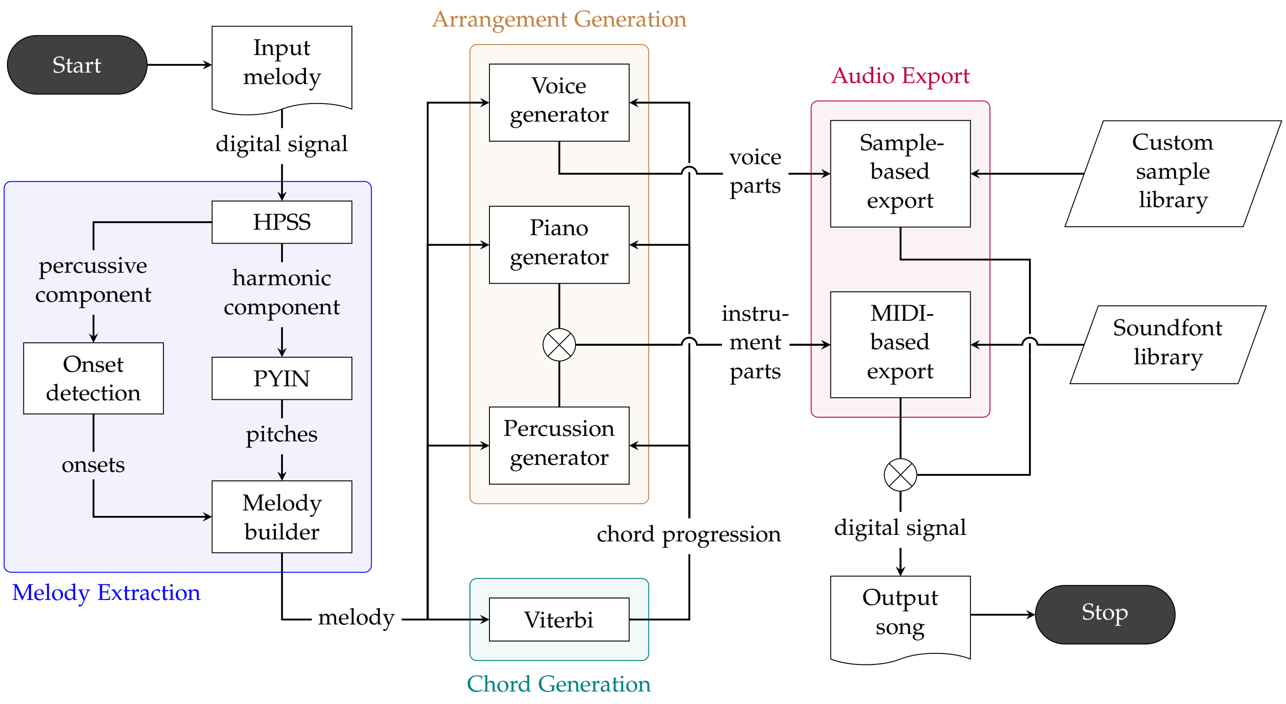


Figure 1: *An overview of the architecture of Piranha Plants as Charade.*

(b) A stride piano part that outlines an appropriate chordal accompaniment.

We consider Piranha Plants as Charade to have adequately met our MVP goals. We also implemented additional features, including exported percussion parts and live deployment in the form of an application programming interface (API).

2 System Overview

Given an input melody in the form of a digital signal, we developed a music generation algorithm that transforms the input with these steps: 1) extract the melodic information from the input; 2) generate a chord progression that fits under both the extracted melody and the appropriate stylistic conventions; 3) generate a musical arrangement based on the melody and chord progression; and 4) export the generated arrangement as a WAV file (see Figure 1).

2.1 Melody Extraction

The first step in Piranha Plants as Charade is to extract the melody from the input audio. The goal of this section is to accept an input audio file and output a sequence of notes, each with a MIDI pitch, start time, and duration. This problem can be broken down into two subproblems: pitch detection and note segmentation.

As a pre-processing step, we first apply the Harmonic-Percussive Source Separation (HPSS) algorithm [2, 3] to separate the harmonic and percussive components of the audio signal. We perform pitch detection on the harmonic component and use the percussive component to identify the timestamps where notes start, which we refer to as note onsets. Finally, we combine both components to produce a sequence of notes to define the melody.

Pitch Detection This problem determines the fundamental frequency of a sound. We aim to produce a sequence of MIDI pitches that correspond to notes in the melody. We explored a few different approaches, starting with cepstrum analysis and autocorrelation. However, these methods failed to handle the complexity of imperfect audio. We elaborate more upon this in Section 4.2. We decided to use the PYIN algorithm [4] as implemented in the librosa[2] library, a state-of-the-art pitch detection algorithm that builds upon autocorrelation to detect pitches more robustly.

The PYIN algorithm is a state-of-the-art pitch detection algorithm that applies autocorrelation to detect pitch candidates and refines the result using a probabilistic thresholding model to filter out spurious frequencies. It is robust on imperfect audio, and has a readily available implementation in the librosa library. After applying PYIN to the harmonic component of the audio signal, we have the estimated

[2] `https://librosa.org/doc/0.11.0/generated/librosa.pyin.html`

pitch in hertz across time. We then apply multiple pitch shifts by a few cents[3] to find the best match, determined by the lowest error after rounding to the nearest MIDI pitch.

Note Segmentation Once we have the pitch sequence, it must be segmented into individual notes. To do this, we count quantized time units from the start of the first identified pitch at the assumed tempo of the song. Within each interval, we take the mode of the pitches as the pitch at that time. We end the previous note and begin a new one if either of the following conditions are met:

1. The pitch changes from the previous time step.
2. An onset is detected in the percussive component of the audio at the corresponding time.

Thus, we obtain a sequence of notes defining the melody to pass to the next component.

2.2 Chord Generation

The next step in Piranha Plants as Charade is to generate the 'best' chord progression[4] for the extracted melody from Section 2.1. It is generally quite open-ended and subjective whether a chord progression sounds good with a melody, and there can be many different appropriate chord progressions that evoke different emotions and styles. For this project, we define 'appropriate' to mean in adherence with the style of "Piranha Plants on Parade". We use a first-order hidden Markov model (HMM) framework to determine the most appropriate chord progression.

The hidden Markov model [5, 6], depicted in Figure 2, is a statistical model that describes a sequence of hidden states, each of which emits an observation. In the context of chord generation, we model the chord progression as a sequence of hidden states and the consecutive notes from the melody as observations. The HMM framework allows us to model the probability of a chord sequence given a melody, which we can then use to generate the most likely or most favourable sequence of chords.

The model consists of the following components for time step $t \geq 0$:

- **States** s_t: The hidden states of the model, which represent the sequence of chords.
- **Observations** o_t: The observations of the model, which represent the melody.
- **Initial State Probabilities (Priors)** $P(s_0)$: The probabilities of starting in a particular state.
- **Transition Probabilities** $P(s_{t+1}|s_t)$: The probabilities of transitioning between states.

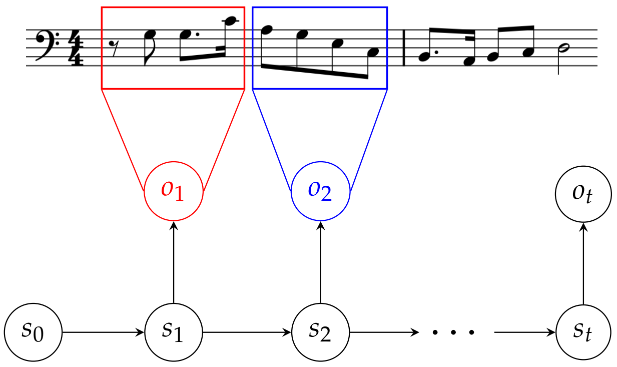


Figure 2: *Hidden Markov model for chord generation.*

- **Observation Probabilities** $P(o_t|s_t)$: The probabilities of emitting an observation given a state.

In the most basic case, states and observations can be mapped to integer indices such that the transition and observation probabilities can be represented as matrices. The model can then be solved using the Viterbi [5] algorithm, which is a dynamic programming algorithm that finds the most likely sequence of hidden states given the observations.

For this project, we implemented the transition and observation probabilities as scores (i.e. unnormalized, relative probabilities). The transition scores represent the favourability of transitioning between two chords, while the observation scores represent the favourability of a melody excerpt given its associated chord. The transition scores encapsulate our analysis of the style of "Piranha Plants on Parade" because said analysis is based on the frequencies of chords and how they flow into one another.

Transition Scores Our transitions are based on an analysis of "Piranha Plants on Parade" from a Western music theory perspective. We identified the following patterns:

1. The I, IV, and V chords[5] and their respective secondary dominant chords[6] almost exclusively used.
2. A secondary dominant chord must be followed by their respective I, IV, or V chord.
3. Secondary dominant chords occur frequently.

In our transition scores, we restricted the valid states to follow the first two patterns, and we applied a 1.5x score multiplier for transitioning away from a secondary dominant chord to model the last pattern.

Observation Scores The observation scores are based on the melody. We calculate the score of a melody given a chord by summing the number of

[3] A logarithmic unit representing a hundredth of a semitone.
[4] A sequence of chords.

[5] Three distinct types of chords. Their composition are irrelevant for this paper.
[6] A type of chord that is dependent on another chord. Its composition is irrelevant for this paper.

notes in the melody that are in the chord. This is a simple heuristic that favours chords that contain more notes from the melody, which is generally a good rule of thumb for harmonization. Although this doesn't fully account for deviations such as passing tones, it provides a good starting point for the model.

2.3 Arrangement Generation

Like many human composers, our arrangement generation process follows a rule-based approach. With the melody and chord progression from Sections 2.1 and 2.2 as context, it generates different parts for each instrument using algorithms conceptualized from a music theory point of view. This is similar to how a jazz musician, who is familiar with the conventions of the genre, can use lead sheets[7] as context to determine what to play in an ensemble. Unlike in this analogy, the different parts cannot 'hear' each other during the generation process, as they are independent of each other. The main advantage of the independent generation approach is its simplicity; however, it does not allow individual parts to influence each other (with the exception of the melody, which can influence all parts); this can result in robotic-sounding parts. For this project, we implemented pipelines that fall under three categories: 1) voice generation; 2) piano generation; and 3) percussion generation.

Voice Generation Generating the melody is trivial: we use the extracted melody from Section 2.1. To generate the vocal harmony line, we use the following algorithm:

> Let S be the set of all MIDI pitches in the C major scale. Let $M = m_1, \ldots, m_n$ and $H = h_1, \ldots, h_n$ be two sequences of MIDI pitches, where m_i and h_i correspond to the ith note in the melody and vocal harmony line, respectively. Compute H as follows:
>
> $$h_i = \begin{cases} m_i - 3 & \text{if } m_i - 3 \in S \\ m_i - 4 & \text{if } m_i - 4 \in S \end{cases}$$

This harmonizes the melody in thirds, which is the same technique applied to most of the vocal harmony in "Piranha Plants on Parade". This algorithm works well for most inputs; however, there is an edge case when the current chord conflicts with the scale. Because the vocal harmony line is quiet, the occasional 'error' does not overly stand out.

[7] A minimalist type of music notation that contains the melody and chord changes in a song.

Piano Generation This is the most complex part of the arrangement generation process. We want to emulate the style of stride piano, which involves alternating between the bassline and the chords. This clear separation of parts allows us to handle the bassline and chords in two separate processes.

Our bassline generation algorithm is straightforward. We alternate between the root pitch of the chord and a perfect fourth below said root pitch (the fifth of the chord), restarting whenever the chord changes.

The chord voicing algorithm is more complex. There are two types of chords we need to support: major (Type 1) chords and secondary dominant 7 (Type 2) chords. It is guaranteed that a Type 1 chord will always follow a Type 2 chord. First, we choose some arbitrary anchor MIDI pitch T and voice all Type 1 chords with the following algorithm:

> Let $C = \{c_1, c_2, c_3\}$ be the set of MIDI pitches of the chord we want to voice. Let $V = \{v_1, v_2, v_3\}$ be the set of MIDI pitches in our computed voicing. Compute V as follows:
>
> $$v_i = c_i + 12 \left(\underset{j \in \mathbb{Z}}{\operatorname{argmin}} |c_i + 12j - T| \right)$$

This step clumps each note in the chord around T, which smoothens chord transitions.

Next, we voice all Type 2 chords with the following algorithm that is based on voice-leading rules in Western music theory:

> Let $M = \{m_1, m_2, m_3\}$ be the voicing of the Type 1 chord following the chord we want to voice. By the definition of a major chord, we can uniquely assign the elements of M such that:
>
> $$m_2 - m_1 \equiv 4 \mod 12$$
> $$m_3 - m_1 \equiv 7 \mod 12$$
>
> Let $V = \{m_1 - 1, m_2 + 1, m_3\}$ be the MIDI pitches in our computed voicing.

Percussion Generation The percussion generation algorithm is very simple: the generated snare drum part outlines a gallop rhythm[8], whereas the bass drum plays on the first beat of every measure. "Piranha Plants on Parade" mostly follows this pattern, thus we include it as a stretch feature in our proof-of-concept.

[8] A repeating pattern that consists of an eighth note followed by two sixteenth notes.

2.4 Audio Export

The final phase of Piranha Plants as Charade is to export the generated song from Part 2.3 as a WAV file, a standard audio format. This step is comprised of two independent parts: 1) a pipeline that uses custom samples to handle the vocal parts; and 2) a pipeline that leverages the MIDI standard to handle all other instruments. The two results are combined to generate the overall output.

MIDI-Based Pipeline By building around the MIDI standard, the bulk of this step is handled by external programs. First, the appropriate song data is written to a MIDI file using MIDIUtil[9], a Python library. Next, the generated file is converted into a WAV file using FluidSynth[10], an open-source audio synthesizer, and the "MS Basic" soundfont[11] by MuseScore. We originally implemented this approach as a prototype, but since it worked well out of the box, we decided to keep this process.

Pipeline for Vocals Although the MIDI-based approach was sufficient for exporting our song as a WAV file, we wanted more customizability than the process allowed for handling vocals. As such, we designed a specialized pipeline to export the vocal parts. We repurposed some voice samples from the video game *Animal Crossing: New Horizons* as the base samples for each voiced syllable; to export a note for a given syllable, we pitch-shifted the respective base sample accordingly. We post-processed each exported note to better fit in the overall audio mix. We first applied a low-pass Butterworth filter to reduce dissonance in the high registers, followed by a volume envelope based on a modified Hamming window to smoothen the start and end of each note.

We pitch-shifted the base samples by hand using Melodyne[12], a commercial software designed for pitch manipulation, and we stored the outputs to be accessed on demand. Although this approach provided us with the highest-quality audio samples, it required a lot of manual work and imposed a cap on the supported pitch range. Nonetheless, audio quality was our top priority in this iteration, so we decided to proceed with the Melodyne approach.

3 Results

Our implementation of Piranha Plants as Charade can successfully take an audio file of the melody as input and generate an audio file of the song in the style of "Piranha Plants on Parade". The generated output contains the melody and a harmony line sung by synthesized vocals, as well as accompanying piano and percussion parts. A collection of sample inputs and outputs is available in our public results repository (See Appendix A.1).

To evaluate the performance of our system, we conducted a qualitative analysis of the generated audio files. We compared the generated output to the original song, "Piranha Plants on Parade", and assessed the following aspects: 1) melody accuracy; and 2) chord progression accuracy.

Table 1: *The melody extraction results over various timbres.*

Timbre		Mistake Count	
Instrument	Noise Level	Pitch	Rhythm
Piano	None	1	1
Piano	Low	1	0
Piano	Medium	1	7
Piano	High	2	9
Flute	None	0	9
Saxophone	None	0	4
Trumpet	None	0	9
Violin	None	Near unrecognizable	

3.1 Melody Extraction Results

To evaluate the efficacy of the melody extraction process, we analyzed the melody extraction outputs for an excerpt from "Piranha Plants on Parade"[13] with the following timbres from MuseScore's "MS Basic" soundfont: flute, piano, tenor saxophone, trumpet, and violin. The pitch and rhythm mistake counts are recorded in Table 1. Additionally, we tested the piano soundfont with different noise levels: no noise, low white noise (0.3% amplitude), medium white noise (0.7% amplitude), and high white noise (1% amplitude).

Qualitatively, the outputs from the tenor saxophone, low-noise piano, and no-noise piano were good. There were errors, but they were few and minor. The outputs for the flute, trumpet, and remaining piano inputs were acceptable; there were clear mistakes, but the melody remains recognizable. The violin output was very poor and bore little resemblance to the input.

Based on the piano data, there is a trend of input noise leading to less accurate outputs. However, the correlation is not perfect; for example, the melody extraction process was more accurate for the piano input with low noise than for the piano input with no noise. Based on the overall data, it is clear that

[9] `https://pypi.org/project/MIDIUtil/`

[10] `https://www.fluidsynth.org/`

[11] A file format that contains instrument sample data.

[12] `https://www.celemony.com/en/melodyne`

[13] The first eighth rest was converted into a note to meet our MVP contraints.

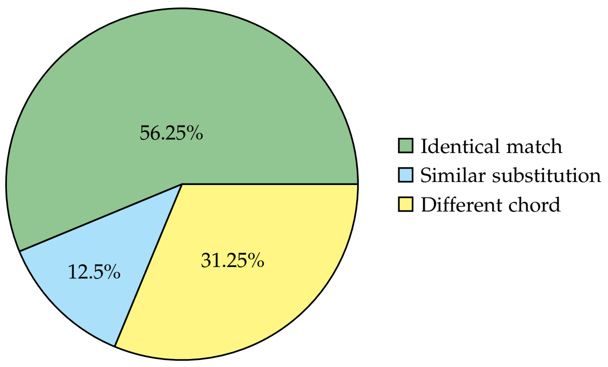


Figure 3: *A comparison between the original "Piranha Plants on Parade" chord progression and that from our chord generation process over the same excerpt.*

the output pitches are generally more accurate than the output rhythms. This implies that the PYIN step of the process is more reliable than the onset detection step because most rhythmic mistakes can be attributed to onset detection errors; extra articulations correspond to false positive onsets and missing articulations correspond to false negative onsets — PYIN errors can only correspond to extra rests (which were uncommon).

3.2 Chord Progression Results

As the first step for evaluating the chord progression generation efficacy, we ran Piranha Plants as Charade with a hard-coded excerpt from the melody of "Piranha Plants on Parade" and we compared the output chord progression to the respective original chord progression.

Overall, the results are very good. The original and generated chords matched perfectly for 9/16 of the excerpt, and a similar chord was chosen for another 2/16 of the excerpt. The chord generation algorithm chooses a different chord only 5/16 of the time; in these cases, the chords chosen by our algorithm seem reasonable. Given that the chord progression generation depends on itself, deviations propagate, thus the output is more accurate than the raw numbers suggest.

Given more time, we planned on running a survey to evaluate the chord progression generation on other melodies. We would have also attempted to find patterns in what melodic traits cause and/or correlate with better or worse chord progression generation.

4 Discussion

4.1 Early Experiments

Cepstrum Analysis Our first attempt at pitch detection involved cepstrum analysis. Our input data consists of a sequence of tones, each of which consists of a fundamental frequency f_0 and harmonics $f_k = kf_0$ for all $k \geq 1$. Thus, we expect the presence of spikes at the multiples of f_0 in the Fourier domain. The idea of cepstrum is to identify this periodic pattern by applying the Fourier transform a second time. Formally, the cepstrum C is defined on a signal y as follows:

$$C = \text{FFT}\left(\log\left|\text{FFT}\left(y\right)\right|\right)$$

The fundamental frequency of the signal is at the first non-zero peak in the cepstrum:

$$f_0 = \frac{1}{\text{argmax}_k\, C\left[k\right]}$$

However, this method is not robust to noise and did not perform reliably in our experiments (See Appendix A.1) using acoustic audio data.

Autocorrelation Another method we explored for pitch detection was autocorrelation. The idea is to find the periodicity of the signal by comparing it to a delayed version of itself. One formulation of the autocorrelation function, adapted from the Pearson correlation coefficient, is given by the following formula:

> Given a signal y and a window size $N \in \mathbb{N}^*$, let y_i be a subsequence of y of length N starting at index i with μ_{y_i} as its mean.
>
> The autocorrelation of a signal with its version delayed by a shift of k is defined as:

$$\begin{aligned} r\left(k\right) &= \frac{\text{Cov}\left(y_n, y_{n-k}\right)}{\text{Var}\left(y_n\right)} \\ &= \frac{\sum_{n=k+1}^{N}\left(y_n - \mu_{y_n}\right)\left(y_{n-k} - \mu_{y_{n-k}}\right)}{\sum_n \left(y_n - \mu_{y_n}\right)^2} \end{aligned}$$

Given the sampling rate f_s, the non-zero shift with the highest correlation between signals corresponds to the signal's fundamental frequency:

$$f_0 = \frac{f_s}{\text{argmin}_k \left\{r\left(k\right) : k \in \mathbb{N}^*\right\}}$$

While this method yielded better results, it was nonetheless limited in handling real audio recordings. The results were very sensitive to hyperparameter choices such as the window size and correlation thresholds. Nevertheless, this served as an interesting exploration into a method that is the foundation for PYIN, the algorithm we decided to use.

Finite State Machine Chord Generation Our first approach to chord generation was to use a finite state machine (FSM) to model the transitions between chords. The FSM is defined by a set of states, representing chords, and a set of transitions, representing the possible transitions between chords. This approach would require us to define transitions individually for each chord, which is a tedious process. Thus, we decided to proceed with a hidden Markov model approach, which allows us to define transitions more easily within two structures: the observation matrix and the transition matrix.

The FSM approach lends itself to the possibility of integrating large language model (LLM) agents, which use FSM-like models to define the decision-making process. However, given our limited testing, LLMs are not able to produce meaningful results when asked to predict the next chord in a sequence given the melody.

Voice Sample Library To generate the vocal parts of our song, we used a custom library of voice samples from the video game *Animal Crossing: New Horizons*. To prepare the library, we needed to pitch-shift the available samples to each note in the scale. Our original attempt used stock DSP algorithms from librosa and other audio processing libraries, but the vocal qualities would become distorted beyond recognition for large shifts. Next, we experimented with formant[14]-preservation techniques, but the results were unsatisfactory due to robotic-sounding artifacts. In the end, we turned to the closed-source software Melodyne to carry on with our MVP. Although we were not able to implement a satisfactory pitch-shifting algorithm, such techniques evidently exist.

4.2 Avenues for Future Work

Relax MVP Assumptions The current implementation of Piranha Plants as Charade is a proof-of-concept that demonstrates the feasibility of our approach. However, it is limited by the assumptions we made to simplify the problem. Thus, we would like to relax these assumptions to expand the range of melodies that can be processed.

For example, we assume that the input melody is played at 110 BPM, which is the tempo of the original song. Moving forward, we would like to step away from this assumption and allow for arbitrary tempos. This would require us to implement a more sophisticated melody extraction algorithm that can handle varying tempos. There are existing methods to identify the beat and infer the tempo, such as BeatNet [7].

We also assume that the key is C major to simplify the chord generation process. Although any melody can be transposed[15] to match this key, as we have done for some sample inputs in this paper, we would ideally like to support any melody in its original key. There are various methods to solve the key-finding problem, such as the Krumhansl-Schmuckler key-finding algorithm [8], an extension of Krumhansl-Schmuckler using intervals [9], and more modern approaches using supervised learning [10].

Improving Melody Extraction Given a simple synthetic melody within our MVP scope, our current implementation of melody extraction works sufficiently well; however, it is not robust enough for recorded audio data. In particular, it does not handle noise well and often has trouble accurately identifying articulations in the melody. Since our onset detection implementation seems to be the source of the most errors, a key point for future work is to refine this step of the pipeline.

Expanding to Additional Styles In the current implementation, the chord progression we generate is determined by the observation and transition models that we define by hand. These capture the cohesiveness of a melody and a chord, and preferences for how to transition between chords respectively. For this MVP, we have only implemented the style of "Piranha Plants on Parade" by specifying a particular instance of the transition and observation models. However, we can easily expand this to other styles by defining new transition and observation models according to different harmonic patterns.

Unfortunately, defining the transition and observation models by hand is a tedious process that requires a deep understanding of music theory. A more scalable method would be to use a data-driven approach to learn the transition and observation models from a corpus of music. This would allow us to tailor the models to a specific style of music programmatically. It is worth noting that a bottleneck for this approach is the availability of data, as it would require a melody and the corresponding chord progression aligned with it.

Extensions of the Hidden Markov Model One key assumption for the first-order hidden Markov model is the Markov assumption: that the next state depends only on the current state, and not the past [6]. While this assumption simplified our approach and provided a simple solution to solve it, this assumption limits the model's ability to capture long-term dependencies, which could be an important element in generating more complex chord progressions. For example, one common cadence[16] appearing in "Piranha Plants on Parade" and across many other gen-

[14]The audio characteristics of spoken sounds.

[15]To change the key of a piece of music by shifting all the notes up or down by a constant interval, maintaining the original contour and harmonic quality.

[16]A harmonic progression signalling the end of a phrase, often creating a sense of resolution.

res, is the ii-V-I cadence. To encourage this cadence, we would need to define a transition model that captures the transitions between three chords. However, this is not possible with the first-order HMM, as its memory is limited to the most recent state. Higher-order HMMs would be able to capture patterns spanning multiple chords. Algorithms for solving higher-order HMMS have been developed, such as the Baum-Welch algorithm [11], a method by Ye and Wang that entails encoding previous states as tuples [12], and an extended Viterbi algorithm for second-order HMMs [13].

Real-Time Processing One interesting expansion of Piranha Plants as Charade is to implement real-time processing of audio input. This would allow for a more interactive freestyle experience, where users can sing or play along with a generated harmony. However, one fundamental flaw of our current approach is that our chord generation algorithm assumes we have access to the full melody to predict the entire chord sequence at once. This is not the case in real-time processing, where we only have access to the melody up to the current time step. Thus, we would need to pivot to a different approach to chord generation that can predict the next chord based on the current melody, and it must be fast enough to do so in real time.

One approach building upon our current work is to process each time step independently, applying the transition and observation models once to determine the next chord. However, this greedy approach would not be able to capture the long-term dependencies of the melody.

We also had the opportunity to discuss our project with Professor Kate Larson[17], who mentioned a recent paper using reinforcement learning for real-time interactive jamming [14]. Although this is outside the scope of our project and knowledge to discuss in detail, it would be an interesting avenue to explore as we consider real-time processing.

5 Conclusion

To summarize, we have presented Piranha Plants as Charade, a system that generates a fully harmonized song from a single melody. Our approach is based on the observation that melodies and harmonies are often closely related, and there are a limited number of ways to change between chords within the "Piranha Plants on Parade" style. We have demonstrated a proof of concept through Piranha Plants on Charade, solving a simplified version of the problem using hidden Markov models, digital signal processing, and more. We have also discussed several avenues for future work, including relaxing the assumptions made in our proof-of-concept implementation, improving melody extraction, and expanding to additional styles. Overall, we believe that our system is an interesting exploration of music generation, especially with the application of a hidden Markov model, and we look forward to continuing to explore its potential.

[17] `https://cs.uwaterloo.ca/~klarson/`

References

[1] Richard. Taruskin. *The Oxford history of western music*. eng. New York: Oxford University Press, 2005. ISBN: 0195169794.

[2] Derry Fitzgerald. "Harmonic/Percussive Separation using Median Filtering". In: *13th International Conference on Digital Audio Effects (DAFx-10)* (Jan. 2010).

[3] Jonathan Driedger, Meinard Müller, and Sascha Disch. "Extending Harmonic-Percussive Separation of Audio Signals". In: *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*. 2014, pp. 611–616. DOI: `10.5281/zenodo.1415226`. URL: `https://doi.org/10.5281/zenodo.1415226`.

[4] Matthias Mauch and Simon Dixon. "PYIN: A fundamental frequency estimator using probabilistic threshold distributions". In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2014, pp. 659–663. DOI: `10.1109/ICASSP.2014.6853678`.

[5] Kate Larson. "Lecture notes for CS 486". 2023.

[6] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models*. 3rd. Online manuscript released January 12, 2025. 2025. URL: `https://web.stanford.edu/~jurafsky/slp3/`.

[7] Mojtaba Heydari, Frank Cwitkowitz, and Zhiyao Duan. *BeatNet: CRNN and Particle Filtering for Online Joint Beat Downbeat and Meter Tracking*. 2021. arXiv: `2108.03576 [eess.AS]`. URL: `https://arxiv.org/abs/2108.03576`.

[8] Robert O. Gjerdingen. "Review: Cognitive Foundations of Musical Pitch, by Carol L. Krumhansl". In: *Music Perception* 9.4 (July 1992), pp. 476–492. ISSN: 0730-7829. DOI: `10.2307/40285567`. eprint: `https://online.ucpress.edu/mp/article-pdf/9/4/476/564999/`

40285567.pdf. URL: https://doi.org/10.2307/40285567.

[9] Søren Madsen and Gerhard Widmer. "Key-finding with interval profiles". In: *International Computer Music Conference, ICMC 2007* (Jan. 2007).

[10] Robert Mahieu. "Detecting Musical Key with Supervised Learning". In: 2016. URL: https://api.semanticscholar.org/CorpusID:38498183.

[11] Leonard E. Baum et al. "A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains". In: *The Annals of Mathematical Statistics* 41.1 (1970), pp. 164–171. ISSN: 00034851, 21688990. DOI: 10.1214/aoms/1177697196. URL: http://www.jstor.org/stable/2239727 (visited on 04/03/2025).

[12] Fei Ye and Yifei Wang. "A Novel Method for Decoding Any High-Order Hidden Markov Model". In: *Discrete Dynamics in Nature and Society* 2014 (Nov. 2014), pp. 1–6. DOI: 10.1155/2014/231704.

[13] Yang He. " Extended Viterbi algorithm for second order hidden Markov process ". In: *9th International Conference on Pattern Recognition*. Los Alamitos, CA, USA: IEEE Computer Society, Nov. 1988, pp. 718, 719, 720. DOI: 10.1109/ICPR.1988.28338. URL: https://doi.ieeecomputersociety.org/10.1109/ICPR.1988.28338.

[14] Alexander Scarlatos et al. *ReaLJam: Real-Time Human-AI Music Jamming with Reinforcement Learning-Tuned Transformers*. 2025. arXiv: 2502.21267 [cs.HC]. URL: https://arxiv.org/abs/2502.21267.

Appendix A: Supplementary Materials

A.1 Project Repository

Project `https://github.com/piranha-plants-as-charade`.

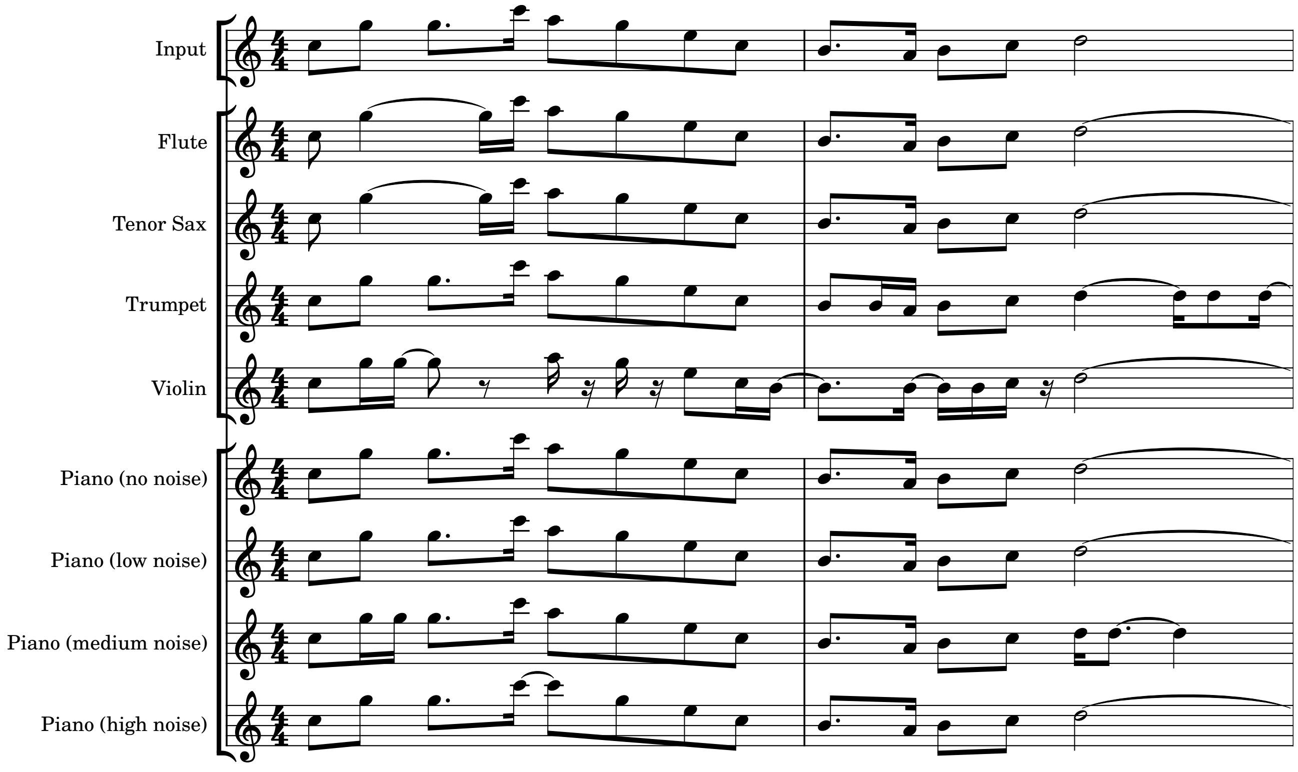
Engine (Main Pipeline) `https://github.com/piranha-plants-as-charade/engine/tree/031c9e3091e4f85bd18fd7f51e3f31b7af37c400`.

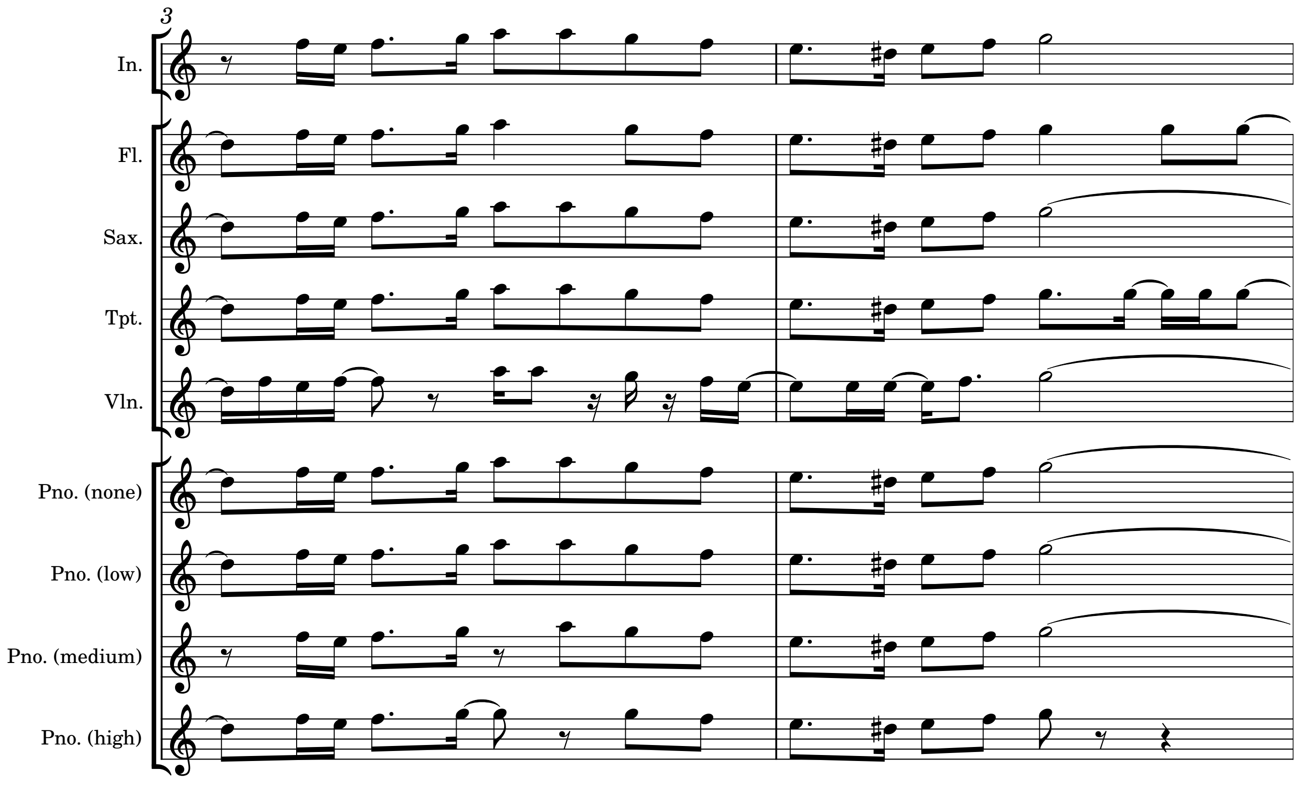
Sample Inputs and Outputs + Melody Extraction Tests `https://github.com/piranha-plants-as-charade/results/tree/066bdf375646d9c8fb28b318f0c3aa2c54683533`.

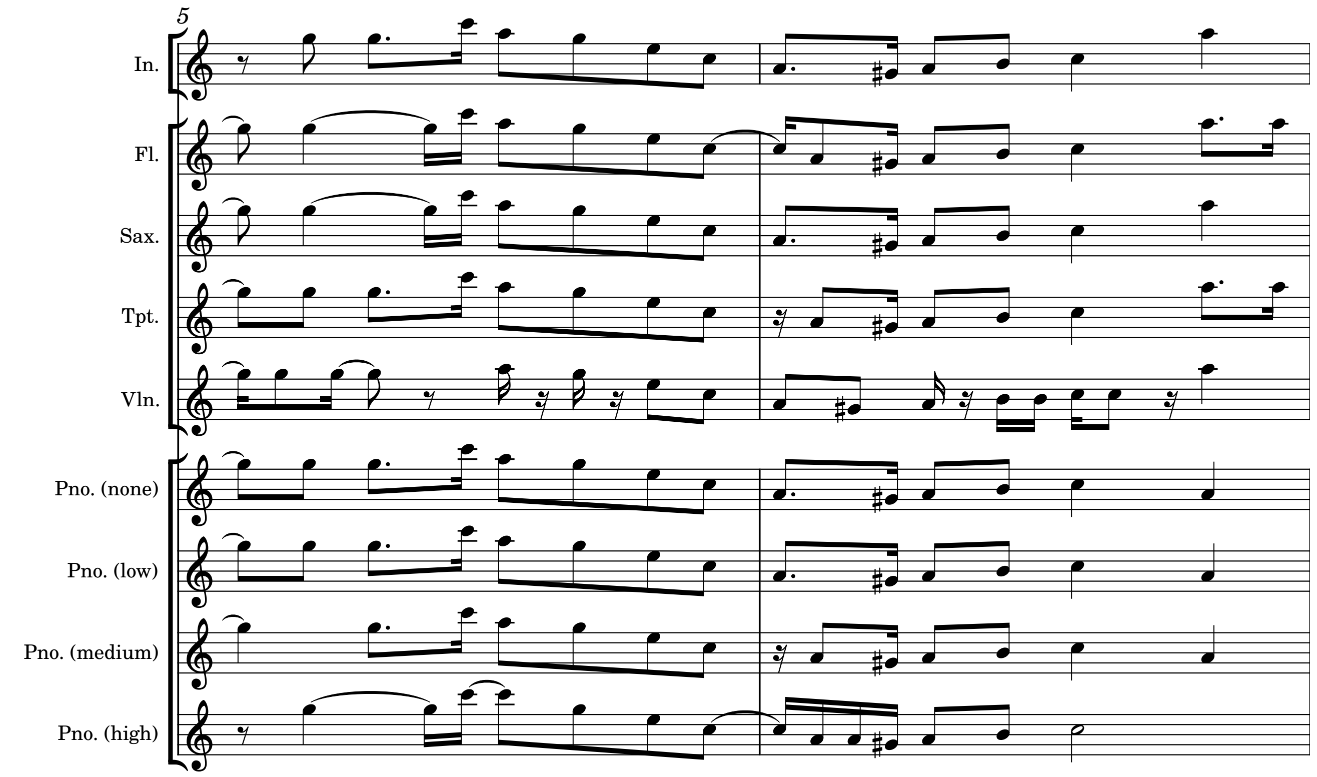
Cepstrum Analysis Notebook `https://github.com/piranha-plants-as-charade/engine/blob/031c9e3091e4f85bd18fd7f51e3f31b7af37c400/playground/cepstrum_pitch_detection.ipynb`.

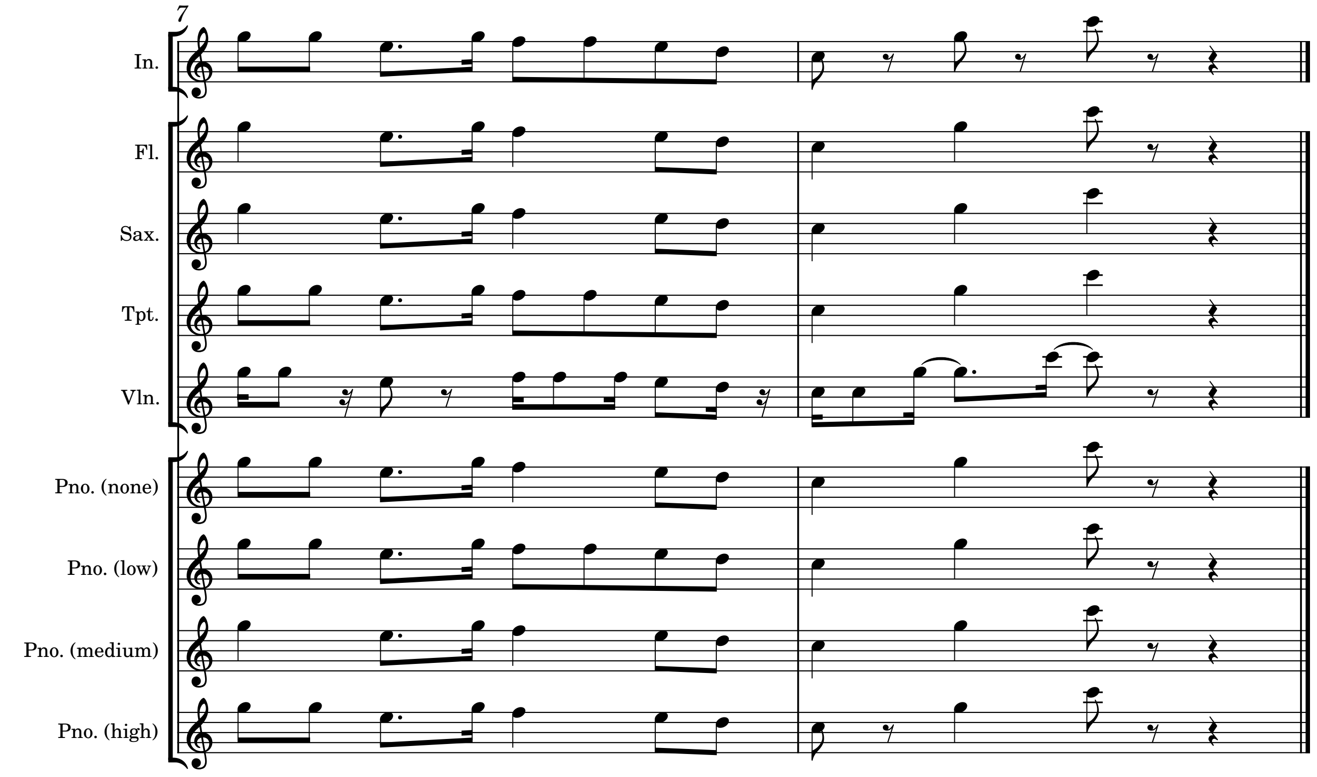
Autocorrelation Analysis Notebook `https://github.com/piranha-plants-as-charade/engine/blob/031c9e3091e4f85bd18fd7f51e3f31b7af37c400/playground/autocorrelation_pitch_detection.ipynb`.

A.2 Transcription of the Melody Extraction Outputs for the Same Melody With Various Timbres

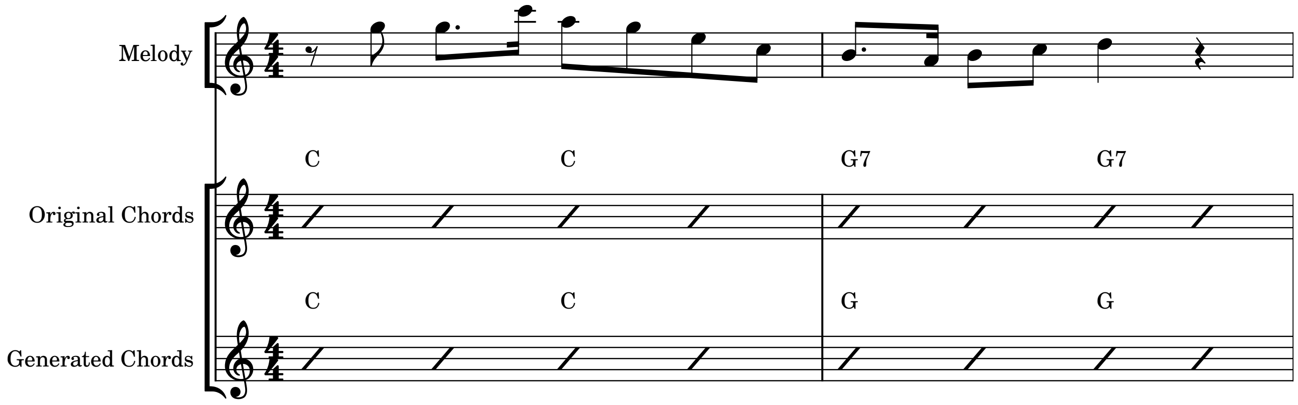


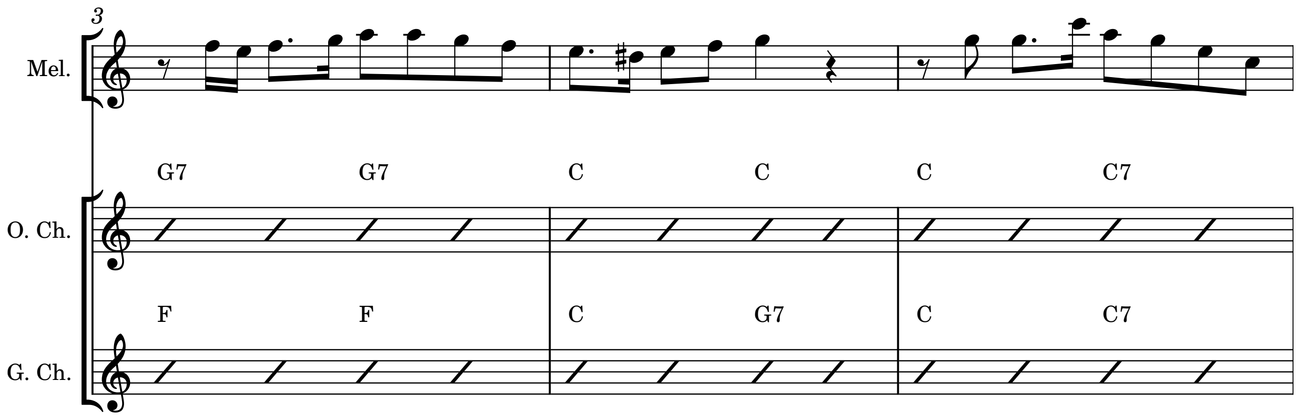


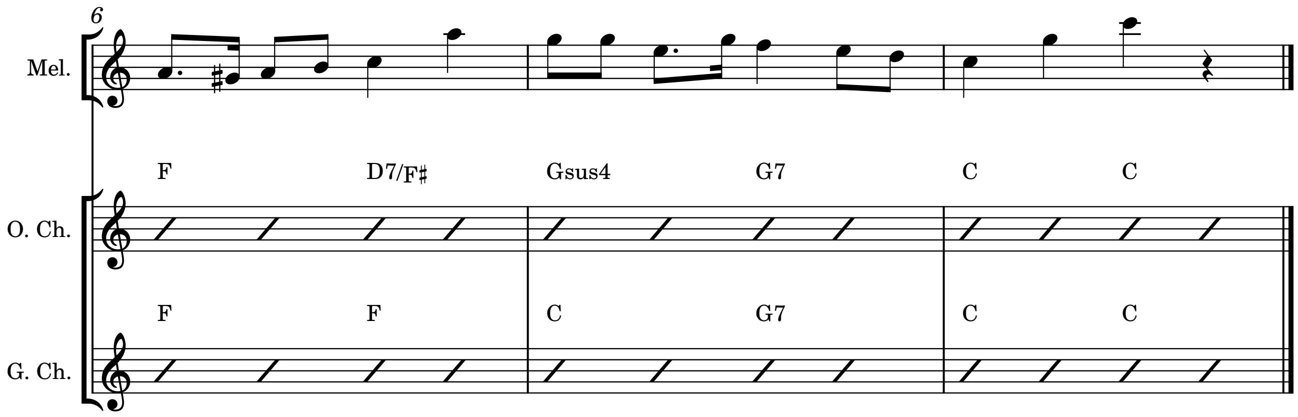

5
In.
Fl.
Sax.
Tpt.
Vln.
Pno. (none)
Pno. (low)
Pno. (medium)
Pno. (high)


7
In.
Fl.
Sax.
Tpt.
Vln.
Pno. (none)
Pno. (low)
Pno. (medium)
Pno. (high)

A.3 Transcription of the Original and Generated Chord Progression for an Excerpt From "Piranha Plants on Parade"


Melody
Original Chords
C C G7 G7
Generated Chords
C C G G


3
Mel.
O. Ch.
G7 G7 C C C C7
G. Ch.
F F C G7 C C7


6
Mel.
O. Ch.
F D7/F♯ Gsus4 G7 C C
G. Ch.
F F C G7 C C