

The Planets

Max Meier

Ludwig-Maximilians-Universität München
max.meier@pst.ifi.lmu.de

Max Schraner

Akademie der Bildenden Künste München
maxschraner@googlemail.com

ABSTRACT

'The Planets' combines a novel approach for algorithmic composition with new human-computer interaction paradigms and realistic painting techniques. The main inspiration for it was the composition 'The Planets' from Gustav Holst who portrayed each planet in our solar system with music. Our application allows to interactively compose music in real-time by arranging planet constellations on an interactive table. The music generation is controlled by painted miniatures of the planets and the sun which are detected by the table and supplemented with an additional graphical visualization, creating a unique audio-visual experience. A video of the application can be found in [1].

Keywords

Algorithmic composition, tangible interaction, soft constraints

1. INTRODUCTION

'The Planets' is an application which allows composing and playing music in real-time without requiring knowledge in music theory. It can be controlled by arranging planet constellations on an interactive table; we implemented it for the Microsoft Surface. Besides playing appealing and diverse music, we also wanted our application to have an aesthetic visual and haptic appearance. The music generation is based on an approach for algorithmic composition where music can be generated in real-time by defining so-called 'preferences' which express 'how the music should sound'. These preferences can also be continually changing, depending for example on user interaction. In our case, the preferences are controlled by planet constellations. This user interface is very abstract and minimal but also gives a high amount of direct control over the music with immediate acoustic and visual feedback.

There is much related work where musical applications are controlled by tangible user interfaces: for example, the famous reacTable [2] provides a completely new paradigm for interacting with a modular synthesizer, Audiopad [3] is based on arranging samples and Xenakis [4] allows composing music with probability models (to name just a few). Most of these applications are geared towards musicians and provide much control but also require a certain amount of musical knowledge. The application presented here particularly has non-musicians as a target group and requires only very basic musical skills. Above all, the work of Toshio Iwai was a great inspiration: Elektroplankton [5] for the Nintendo DS handheld gaming

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

NIME2010, 15-18th June 2010, Sydney, Australia
Copyright remains with the author(s).

console offers a collection of several musical mini-games and allows creating music in a very playful way and Yamaha's Tenori-On [6] provides a matrix of 16x16 lighted buttons which are used to control several intuitive music generation applications.

The employment of planets (which have no direct connection to music) was inspired by the orchestral suite 'The Planets' by Gustav Holst: every planet in our solar system is portrayed with a musical piece which reflects its special astrological character. The earth is not included in Holst's composition so we also decided to omit it.



Figure 1 The Planets

In the next section, we will introduce our application from a user's point of view. Then, we will take a closer look at how we realized the tangible objects, the table visualization and the music generation. Finally, we will draw conclusions and indicate further work.

2. CONCEPT

The Planet's user interface is entirely based on miniatures of the sun and the planets in our solar system which can be arranged on an interactive table and generate music depending on their current constellation. A planet's absolute position on the table does not play a role – only its relative position towards the sun is of importance. There are five smaller planets (Mercury, Venus, Mars, Neptune and Uranus) each representing an instrument with a different sound. Moving an instrument planet towards the sun makes it play faster (more and shorter notes); a planet which is far away from the sun plays only few notes.

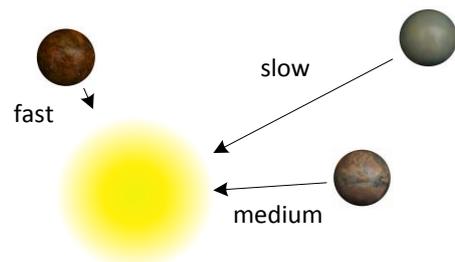


Figure 2 Speed

The instrument's pitch is controlled by the planet's relative angle to the sun: rotating it clockwise around the sun increases its pitch, rotating it counter-clockwise decreases it such that a full rotation corresponds to one octave. The pitch is not controlled deterministically; there are always a larger number of possible pitches corresponding to a certain angle. Whenever a planet plays a note, a supplementing visualization is displayed on the table: at the position where the note was started, a sphere appears which becomes bigger and fades out over time. The visualization does not follow the planet but rather stays where the planet was when the note was started. This way, a planet being moved on the table leaves a trace of spheres behind it representing the notes it recently played.

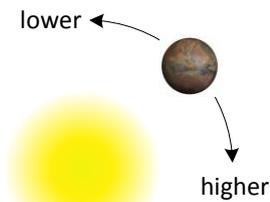


Figure 3 Pitch

Jupiter and Saturn do not play notes – instead, they control global parameters affecting the interplay between the instrument planets; they are also bigger than the other planets in order to make their special role clear. Jupiter controls the global 'harmony' between all instrument planets: the closer it is to the sun, the more harmonic intervals between the instrument planets are played (like fifths, fourths or thirds). This is also being reflected in its visualization on the table: when there is high harmony (Jupiter is near the sun), its visualization is green – moving it away fades its color to brown ('medium harmony') and, finally, to red ('no harmony at all'). Rotating Jupiter around the sun changes the global tonal scale. This is done in steps of fifths in order to create natural harmonic modulations around the 'circle of fifths'. For example, when the current tonal scale is c-major (also corresponding to a-minor in our case, since there is no fixed tonic), a clockwise rotation modulates to g-major (and then to d-major, a-major and so on). Playing just notes from a given tonal scale is only enabled when Jupiter is within a certain range around the sun – if it is out of this range, there is no restriction to a scale anymore and any note can be played.

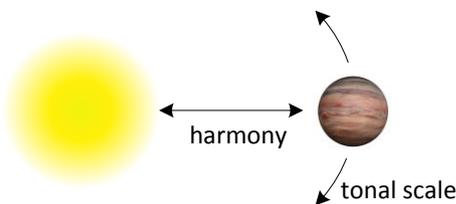


Figure 4 Jupiter ('Harmony')

Saturn controls global rhythmic parameters. When Saturn is near the sun, the global rhythmic accuracy is high and every metric time interval has a constant length (e.g. any 16th is as long as any other 16th). Moving it away from the sun leads to a more loose and imprecise rhythm with random tempo variations. The global tempo can be controlled by moving Saturn clockwise (faster) or counter-clockwise (slower) around

the sun. Saturn's visualization on the table reflects both parameters it controls: Every 8th note, it emits a circle which becomes bigger and fades out. This visualizes both the tempo and the rhythmic accuracy: the faster the tempo, the smaller the distance between the circles; a high rhythmic accuracy leads to circles with equal distance.

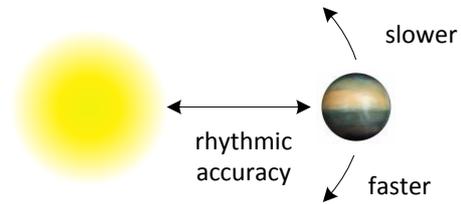


Figure 5 Saturn ('Time')

3. IMPLEMENTATION

We implemented our application for the Microsoft Surface table which recognizes fingers and objects put upon its display. This makes it possible to use new interaction paradigms based on multiple fingers ('multitouch') and dedicated physical control objects ('tangibles') which are detected either by their form or by a visual tag.

3.1 Tangibles



Figure 6 Jupiter Tangible

The planet miniatures are realized as half-spheres made of aluminum. We decided to use this material because it is very robust and has a good haptic quality on the one hand (in contrast to e.g. wood) but still is not too heavy on the other hand (in contrast to e.g. steel). We wanted the tangibles to look like the real planets in our solar system; they are painted using techniques from the area of 'trompe l'oeil'-painting. 'Trompe l'oeil' can be translated as 'trick the eye' and tries to make a painting look like a real thing. It can often be found on the facades of buildings, adding e.g. fake windows or pretending the use of expensive material (like marble). Our tangibles are painted precisely based on satellite images of the real planets. Although being relatively small with diameters of only 5 and 6 cm, many realistic details are captured on them. Since the tangibles are meant to be touched and played with, they are covered with an additional protective layer on top of the painting. For realizing the sun, we wanted to make use of a transparent material in order to illuminate it using the table's display. Real glass has a very good haptic quality – but it is not robust enough and would also be too heavy for a half-sphere with a diameter of 8.5 cm. We finally decided to use acrylic glass which is sandblasted on the spherical side, thus creating a diffuse texture. The sun's flat bottom side is not sandblasted in order to let as much light as possible pass through it.

3.2 Visualization

The visualization on the table is designed to look appealing but also to help in understanding how the system works and give additional visual feedback to acoustic events. We implemented it in .NET using WPF (Windows Presentation Foundation) and the Microsoft Surface SDK which provides several table-specific WPF controls.

The main design elements are spheres. The sun is the only tangible object with has a static sphere being constantly displayed below it, filled with a radial gradient fading out to transparency. All other objects (the planets) only emit spheres which stay at their original position. An instrument planet does not have a constant visualization: only when it plays a note, it leaves a sphere at the position where the note was started which becomes bigger and fades out over time. The sphere does not follow the planet but rather stays in its initial position. This way, a planet being moved on the table leaves a trace of spheres behind it, visualizing the notes it recently played.

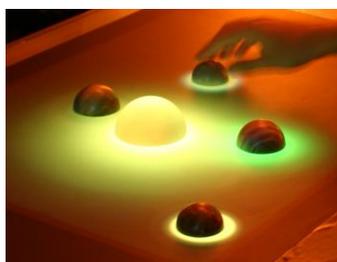


Figure 7 Table display

Jupiter, controlling the global harmony between the planets, emits a filled sphere every 8th note. The sphere's color visualizes the current harmony: when there is high harmony, its color is green. With decreasing harmony, the color continually fades to brown representing 'medium harmony' and then to red, representing 'no harmony at all'. Similarly, Saturn (controlling rhythm) also emits a sphere every 8th note. These spheres are not filled and should resemble Saturn's rings. The distance between these rings visualizes the global tempo (small vs. large distance) and the rhythmic accuracy (equal vs. irregular distance).

3.3 Music Generation

Our application is based on a novel technique for algorithmic composition. We use a framework [7] which allows generating music in real-time by defining certain 'preferences' that express 'how the music should sound'. As an example, a preference for a single instrument could be 'fast notes with a high pitch'. Besides preferences for single instruments, it is also possible to define preferences for the coordination of multiple instruments. Coordination preferences typically involve harmonic or rhythmic aspects, for example 'play together in a similar rhythm in rather consonant intervals'. Most of these preferences are rather 'soft': they do not define an exact result and do not need to be strictly obeyed all of the time. In fact, there are often preferences which are even concurrent among each other; for example, a single instrument's rhythmic preferences could be in conflict with global rhythmic preferences. In our application, each instrument states its own preferences for notes, depending on the corresponding planet's relative position to the sun. These preferences do not already determine a certain note, but rather define a larger possibility space for notes. All instruments' preferences are combined and extended with global preferences. An important global preference states the desired global

harmony (controlled by Jupiter): When Jupiter is near the sun, harmonic intervals between the single instruments are preferred. The weight of this preference decreases along with Jupiter's distance to the sun.

Preferences are expressed as 'soft constraints' which allow dealing with soft and concurrent problems in an easy way. Bistarelli et. Al. developed a very elegant and abstract theory of soft constraints [8]; the algorithmic composition framework [7] is based on an extended theory [9] which additionally allows defining 'meta-preferences' over the constraints itself. Constraint programming in general is a declarative programming paradigm: instead of defining a concrete algorithm which computes a desired result step-by-step, the result itself is described by several 'properties' it should have; a general solver can then be used to compute the result. In classical constraint programming, a desired result is specified with several conditions constraining the properties of solutions. These conditions are often expressed as logical formulas which imply a hard border between correct and incorrect solutions. Classical constraint satisfaction problems can model many different problems ranging from cabin layout design for airplanes to proving correctness of software. However, there are also many real-life problems where classical constraints come to their limits because the border between 'good' and 'bad' solutions is fuzzy or hard to formulate. A way to deal with such problems is soft constraints, which are a generalization of classical constraints. Instead of drawing a hard line between solutions and non-solutions, soft constraints rate every alternative by assigning a grade to it.

We will now give a short and simplified introduction to soft constraints and how they can be used to generate music (for more detailed information, we refer to the cited works). In general, we model a problem as an assignment of values to variables (a valuation) under certain conditions: for example, we could model a scheduling problem by assigning tasks to time slots; in our case, we want to assign notes to planets. Valuations are defined as functions from variables to values: $Variable \rightarrow Value$. Soft constraints rate valuations by assigning a grade to them; this way, an order over valuations is induced. A soft constraint is a function from valuations to grades: $(Variable \rightarrow Value) \rightarrow Grade$; many different types of grades can be used, for example numbers or Boolean values. Several properties or optimization goals of a desired valuation can then be described by soft constraints. These can be combined in various ways and a general solver can be employed to compute the 'best' valuation which yields the highest grade. When it comes to generating music in our application, we want to assign 'actions' to the planets. We use three different types of actions: start a note with a given pitch, hold a note or pause. The desired action assignments are described by soft constraints. At certain time intervals (each 16th note, the speed and rhythmic accuracy is controlled by Saturn), a soft constraint problem is generated based on the current planet constellation. This is being solved and the resulting actions are performed. Based on a planet's relative position to the sun, one soft constraint describes its current preferences for pitch and 'rate of notes'. For example, when it is far away from the sun, the pause action will get a rather high grade compared to the note actions; rotating it around the sun shifts the grades to other note pitches. Each planet states its own preferences with a soft constraint $planetConstraint_{pePlanet}$; these are combined with an additional constraint which optimizes the harmony between the planets. This is realized using a function which rates the harmony between the note pitches of two actions $harmony : Action \times Action \rightarrow Grade$,

returning high grades for consonant intervals (e.g. a fifth) and, vice-versa, low grades for dissonant intervals (e.g. a tritone). Given a set of several planets, a constraint maximizing the harmony in this set can easily be defined by just summing up the harmony between all pairs of planets; the constraint solver will do the rest and solve the task of optimizing this function:

$harmonyConstraint : (Planet \rightarrow Action) \rightarrow Grade$

$$harmonyConstraint(val) = \sum_{p_1 \in Planet} \sum_{p_2 \in Planet} harmony((val(p_1), (val(p_2)))$$

Depending on Jupiter's distance to the sun, this constraint is weighted with a factor *jupiter* which becomes greater the closer Jupiter is to the sun. The final constraint problem is then defined by adding up the weighted harmony constraint and all planets' constraints:

$thePlanets : (Planet \rightarrow Action) \rightarrow Grade$

$$thePlanets(val) = jupiter * harmonyConstraint(val) + \sum_{p \in Planet} planetConstraint_p(val)$$

To sum it up, for each planet $p \in Planet$ we define a constraint $planetConstraint_p$, expressing its preferences for pitch and 'rate of notes'. An additional constraint $harmonyConstraint$ optimizes the harmony between the planets, weighted with a factor *jupiter* depending on Jupiter's distance to the sun.

When the soft constraint problem has been solved, the solution's actions can be sent to a sound generator which transforms the notes to audio, for example a software synthesizer running on the Surface or MIDI equipment.

4. CONCLUSION

Soft constraints showed up to be very appropriate for composing music in our application. Most code in the implementation deals with the graphical user interface and technical details (e.g. tracking planet rotations around the sun); only few lines of code were necessary for realizing the music generation itself: the implementation of a typical constraint like the 'harmony constraint' only takes about 10 lines of code.

We made the observation that it is hard to generate music fitting to a certain musical 'style' (e.g. Jazz or Country music). Manually identifying and defining the preferences for a musical style is a lot of work. We are currently investigating ways of automatically extracting melodic preferences for a given style by analyzing existing melodies.

Furthermore, we are also very interested in developing new interaction paradigms and applications for interactively composing music in real-time based on soft constraints.

5. ACKNOWLEDGMENTS

Many thanks to Prof. Reif for providing his Surface table and Andreas Angerer for helping in working with it!

6. REFERENCES

- [1] The Planets video <http://vimeo.com/planets>
- [2] S. Jordà, G. Geiger, M. Alonso and M. Kaltenbrunner. The reacTable: Exploring the synergy between live music performance and tabletop tangible interfaces. *In Proceedings of the 2007 conference on Tangible and embedded interaction.*
- [3] J. Patten, B. Recht and H. Ishii. Audiopad: a tag-based interface for musical performance. *In Proceedings of the 2002 conference on New interfaces for musical expression.*
- [4] M. Bischof, B. Conradi, P. Lachenmaier, K. Linde, M. Meier, P. Pötzl and E. Andre. XENAKIS - Combining tangible interaction with probability-based musical composition. *In Proceedings of the 2008 conference on Tangible and embedded interaction.*
- [5] T. Iwai, Indies Zero and Nintendo. Electroplankton. *Music Game for Nintendo DS, released in 2005*
- [6] Y. Nishibori and T. Iwai. Tenori-On. *In Proceedings of the 2006 conference on New interfaces for musical expression.*
- [7] M. Hölzl, G. Denker, M. Meier and M. Wirsing. Constraint-Muse: A Soft-Constraint Based System for Music Therapy. *In Proceedings of the 2009 conference on Algebra and coalgebra in computer science.*
- [8] S. Bistarelli, U. Montanari, F. Rossi. Semiring-based constraint satisfaction and optimization. *In Journal of the ACM, Volume 44, Issue 2 (March 1997) Pages: 201 - 236*
- [9] M. Hölzl, M. Meier and M. Wirsing. Which Soft Constraints do you Prefer? *In Proceedings of the 2008 Workshop on Rewriting Logic and its Applications.*