SolAR: exploration into spatial data sonification

Candidate Number:	196484
Module Name:	Composing for Media
Module Code:	828W3
Academic Year:	Autumn 2018

Introduction

SolAR, in essence, is an exploration into a possible use case for mixed reality technology that shies away from the norm of **visually** augmented gaming and utility applications. I developed this application because I believe that it has the potential to scale into multiple settings, including networked interaction and installation models. With modern advancements in AR technology, i.e, the release of Apple's ARKit 2.0 - it makes it more accessible for developers to create complex augmented and mixed reality applications.

In its current form, SolAR is a spatial audio experience where the user is the centre of a randomly generated or user-parametrised solar system. Through data sonification, celestial bodies orbiting the centre of the system are heard orbiting the user at variable paces. Their audio source is a real-time synthesiser that creates a random sequence based on a predetermined scale. Parameters of the synthesiser are set at run time based on the respective physical properties of the individual celestial objects.

My primary motivations for the developing of this software is my fascination of the cosmos, the development of modern AR technology, and my interest in aurally perceiving information and object properties through data sonification.

To access the commit history of the project on GitHub, navigate to:
https://github.com/sambilbow/SolAR/commits/master

Research Background

Data sonification in creative practice

I would like to preface this section by discussing two concepts: 'added value', and 'data sonification'.

Added value

In the field of audio-visual composition and performance, Michel Chion notes the concept of 'added value' - the idea that the total effect of synchronised audio and visual elements is greater than the individual effects of separated audio and visual elements (Chion 1990); if one element was taken away, the total effect is less impactful.

Data sonification

Data sonification is defined as the use of non-speech audio to convey information (Kramer, 1994). There are a number of ways in which this information can be utilised aurally, with well known examples including medical (heart rate monitors), military (cockpit auditory displays and sonar) and automotive (speedometer alarms) uses. The use of auditory displays is especially important when we consider how reliant consumers of technology are on visual display of information. Including auditory display has the benefit of increasing the channels of communication between human and computer, which increases bandwidth (speed of information consumption), and decreases visual overload.

When considering data sonification as a method of composition within creative practice, it must be taken into account that the sonification exists of the data, but the data not of the sonification. In most formats it is a one way process, which, importantly, has been defined by the creator of the system and is **arbitrary.** When one replaces the term 'visual' with 'informational', I believe that Chion's concept of added value exists within this context - in that the value of the sonification is wholly attributed to the **process** (Figure 1) of the sonification, and as a system, there is added value to the data.

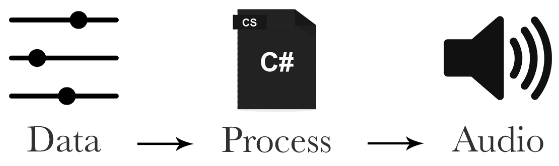


Figure 1: Sonification process

Examples of contemporary works of data sonification that have influenced SolAR include Ryoji Ikeda's *datamatics* (Ikeda, 2006). In this audiovisual project, Ikeda uses software to aurally perceive otherwise invisible datasets. It has taken many forms, including audiovisual concerts, installations, and CD releases. During the development of SolAR I came across Nick Ryan's installation work *Machine 9.* Machine 9 is a physical instrument that tracks the position of 27,000 pieces of space debris, and transforms their locational data into sound as they pass over Earth. Machine 9 was engineered to make people more aware of the dangerous amounts of this debris by “giving them voice” (Ryan and Le Couteur, 2016).

AR in creative practice

The advancement of mobile AR technology being focused on camera technology (namely Apple’s ARKit and Google’s ARCore) has lead to visual displays being the main form of AR consumption (Kiefer and Chevalier, 2018). The majority of AR experiences depend, therefore, on the user’s handheld device being the ‘viewfinder’ into the mediated ‘reality’ that they are to experience. Not only does this reduce immersion, it has also lead to vast amount of “AR” applications that don’t ‘augment’ anything other than the handheld device’s reality, let alone the user’s.

Whilst I believe that the implementation in handheld devices of AR is positive due to its now widespread use and development, it has been damaging for the use of creative practitioners using AR as a medium for computational art to sideline other sensory aspects of a users ‘reality’ such as auditory and haptic perception - especially when these modes of sensory perception were being discussed at the same time as using visual perception modes (Azuma, 1997). Perhaps the industry will see a shift back to head-mounted displays, allowing for more immersive multi-modal experiences. The main limitation in doing this would be the social acceptance of mainstream head-mounted displays in every day life, which would depend on the unobtrusiveness of the system, as well as addressing with privacy concerns of using potentially hidden camera technology in public (Krevelen, 2010).

In SolAR, I hope to start a project that has a wide range of future uses for its data sonification, implementing networked AR experience into the app is definitely in the app’s development roadmap.

SolAR

Design

SolAR aims to allow the user to perceive inaudible celestial objects aurally, using a mix spatial audio techniques and AR technology. It is currently for iOS smartphone users as it has been developed using ARKit, and is ideally experienced with high quality stereo headphones. In its current iteration, the software is more of a relaxation virtual reality experience, where imaginary solar systems are generated from user specified or random numbers, however future iterations could see SolAR being an educational tool that uses real-time data from our own solar system to visualise and sonify various natural and human-made celestials with a head-mounted display such as the Microsoft HoloLens.

SolAR is developed in Unity3D due to its engine natively allowing spatial audio. It makes use of the ARKit plugin, which has allowed me to implement a system that takes the users position into consideration. With the release of ARKit 2.0 allowing networked AR interactions within the same 3D world, I thought that it would be a good idea to use this technology as it will allow me to implement more advanced features into SolAR in the future including the possibility of an installation where users can interact and and compose within a generated SolAR system.

The limitation of using a handheld device such as a smartphone for this project is that in stereo spatial audio, sound sources can only be perceived in one dimension, the X axis. However, in SolAR, a second dimension can be perceived due to Unity3D's "3D sound settings". With this parameter, I can choose to have the 3D engines information about how far away a sound source is from the listener attenuate the volume of the sound source. This allows the perception of the Z axis, or the distance away from the listener laterally. The advantage of developing SolAR in Unity3D is the ability to render the software to different operating systems. This means that I could use the app on a device with more than 2 channel outputs, achieving a more immersive experience at the cost of portability.

Implementation

In the early development of this software, I was experimenting with placing 3D objects in space using ARKit, the user had a visual display of these objects and could place, as well as displace, objects by colliding them with each other. At some point during the development of this 3D placing software, I had the idea to instance celestial objects instead, that rotated around the user. This lead to me creating a 'prefab' for a default celestial object (prefab is essentially a term for a template object). Upon researching the use of AR in computational art, I decided to remove the visual aspect of reality mediation, and focus on the aural display, to see just how immersive I could create a spatial system using one sensory mode. ARKit was instrumental in improving the immersion of the system as it takes gyroscopic data from the users phone and uses it to position them in the system rotationally. This means that when you rotate in your own environment, you can hear the relative positions of the celestials move (as well as their own orbital movements).

SolAR operates based on object-oriented programming through combination of authored 3D objects, and a number of C# scripts that I have developed. There are currently two modes of use for the app - 'generate system' and 'create system'. The main premise of the scripts I have written are to instance the prefab objects, and then assign those objects individual properties by using *for loops*. Four of these properties then set parameters on the synthesiser engines attached to each object, and a generate sequence script generates a sequence based on a predetermined scale. An audio manager object with a script attached automatically assigns objects (after they have been instanced) to mixer groups that have the AudioHelm synth attached to their channel strips. This script flow is shown in figure 2.

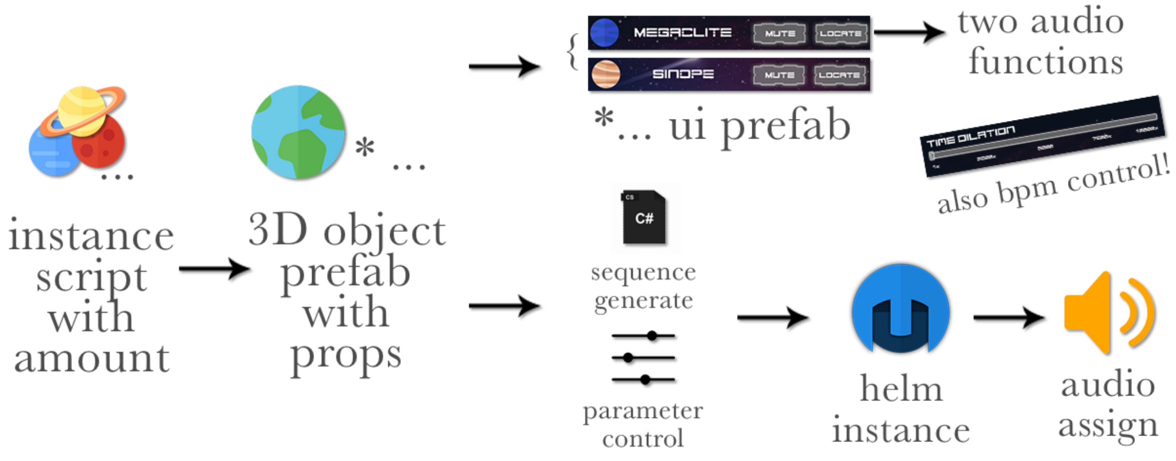
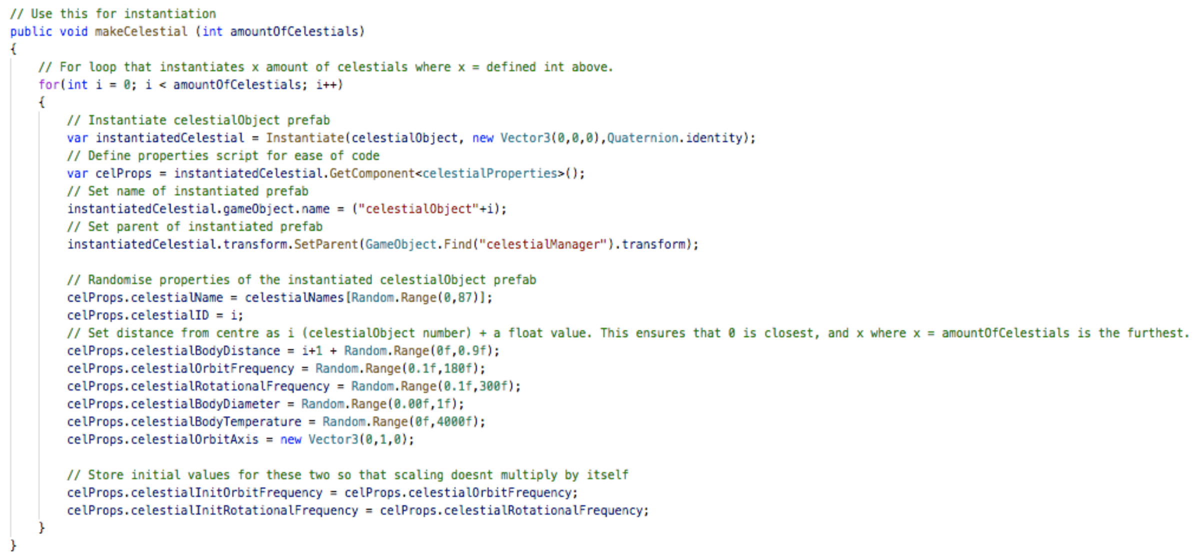


Figure 2: Sonification of data within SolAR

Coding4Fun (Microsoft) Design Engineer, Rick Barraza's video series on creative coding with Unity was instrumental in my understanding of the idea of coding procedurally - creating templates with property values that could be cloned and altered with 'master/manager' scripts (Barraza, 2016).

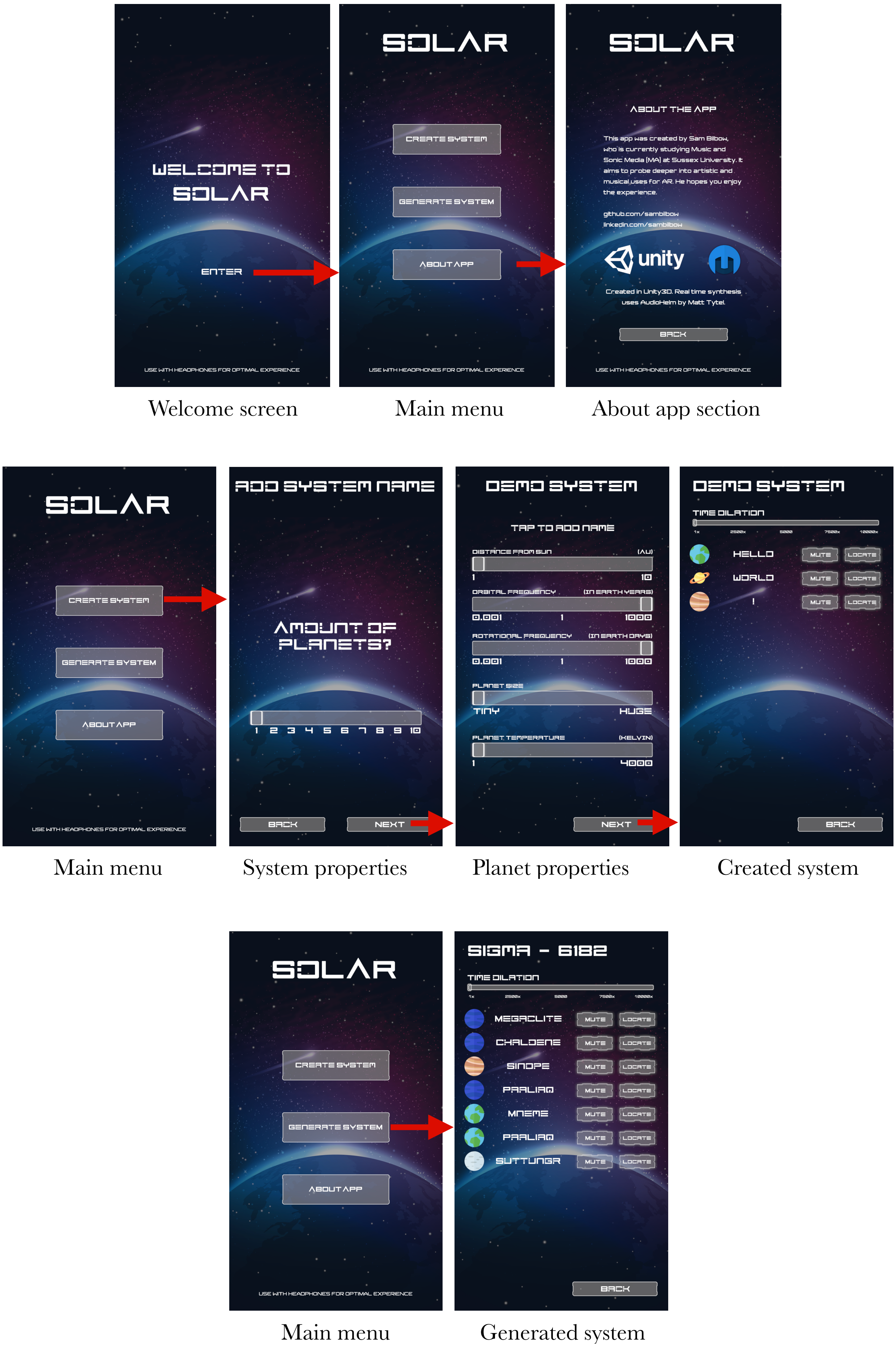


```
// Use this for instantiation
public void makeCelestial (int amountOfCelestials)
{
    // For loop that instantiates x amount of celestials where x = defined int above.
    for(int i = 0; i < amountOfCelestials; i++)
    {
        // Instantiate celestialObject prefab
        var instantiatedCelestial = Instantiate(celestialObject, new Vector3(0,0,0),Quaternion.identity);
        // Define properties script for ease of code
        var celProps = instantiatedCelestial.GetComponent<celestialProperties>();
        // Set name of instantiated prefab
        instantiatedCelestial.gameObject.name = ("celestialObject"+i);
        // Set parent of instantiated prefab
        instantiatedCelestial.transform.SetParent(GameObject.Find("celestialManager").transform);

        // Randomise properties of the instantiated celestialObject prefab
        celProps.celestialName = celestialNames[Random.Range(0,87)];
        celProps.celestialID = i;
        // Set distance from centre as i (celestialObject number) + a float value. This ensures that 0 is closest, and x where x = amountOfCelestials is the furthest.
        celProps.celestialBodyDistance = i+1 + Random.Range(0f,0.9f);
        celProps.celestialOrbitFrequency = Random.Range(0.1f,180f);
        celProps.celestialRotationalFrequency = Random.Range(0.1f,300f);
        celProps.celestialBodyDiameter = Random.Range(0.00f,1f);
        celProps.celestialBodyTemperature = Random.Range(0f,4000f);
        celProps.celestialOrbitAxis = new Vector3(0,1,0);

        // Store initial values for these two so that scaling doesnt multiply by itself
        celProps.celestialInitOrbitFrequency = celProps.celestialOrbitFrequency;
        celProps.celestialInitRotationalFrequency = celProps.celestialRotationalFrequency;
    }
}
```

Figure 2: celestialObjectInstantiator.cs

Figure 3 shows the for loop that clones an amount of celestial objects and assigns random property values to them. Some values based on the ID of the clone, which allows for patterns such as the first clone being the closest to the centre. This script is used in the *generate system* mode.

The *create system* mode differs in that it allows the user to specify all of the property values for the system. The user can control the amount of objects and their parameters. I had started working on an actual solar system model before starting the development of this create system mode, until I realised that it was a step in the wrong direction. I realised that I shouldn't be hand-crafting the solar system in a 3D engine, instead, I should be making the template scripts and objects that would allow me to simply input an xml or other similarly formatted property file that could get parsed and do all of the work for me - thats what procedural programming is all about after all, right? It was at this point in the project where I devoted a lot of time into the create system. I had a specific UI flow in mind, which is shown in the middle section of Figure 4 below. It was after the creation of this UI in Unity that I came across a bug where the software would crash on iOS when using the *create system*. After meticulously scrutinising my code for two days and rewriting most of it (which was a great exercise in identifying my own bad coding habits), I realised that the way that I was referencing gameObjects in a way that iOS couldn't understand, and as a result, the celestials weren't being instanced - which was crashing the whole app. This isn't documented anywhere in the Unity API, hence why it took a while to debug.



This app was created by Sam Bilbow, who is currently studying Music and Sonic Media (MA) at Sussex University. It aims to probe deeper into artistic and musical uses for AR. He hopes you enjoy the experience.

github.com/sambilbow
linkedin.com/sambilbow

Created in Unity3D. Real time synthesis uses AudioHelm by Matt Tytel.

Figure 4: Flow of UI

Evaluation

At the conception of my idea of SoLAR, I sought to create a a spatial audio experience with data sonification, I believe that SolAR achieves this, and lays the foundation for a greater experience to be developed in the future. A lot of the challenges associated with this project were attributed to learning C#, and how to think computationally about generative art - for example realising when to create templates, and what makes a good creative mapping in terms of data sonification.

In the future, I definitely plan to develop the audio system in SolAR, by fine tuning the sequence generator (which is modified from a demo in the AudioHelm documentation) to include a Markov-like system based on interactions within the generated system (i.e. distance between celestials using Unity's raycast function) to generate more sonically rich textures and sounds. In addition, I would enjoy taking SolAR into the context of a sound installation, implementing ARKit 2.0's networked AR capability to create shared composition experiences. I envision this experience to be in a multichannel format much more advanced than stereo or even quad channel - implementing the ambisonic format into SolAR would improve group and individual immersion by a great deal.

Biblography

Azuma, R. (1997). A Survey of Augmented Reality. *Presence: Teleoperators and Virtual Environments*, 6(4), pp.355-385.

Barraza, R. (2016). *Creative Coding with Unity | Channel 9*. [online] channel9.msdn.com. Available at: https://channel9.msdn.com/Series/UnityCreativeCoding [Accessed 15 Nov. 2018].

Chion, M. (1990). *Audio-vision*. New York, NY: Columbia Univ. Press.

Ikeda, R. (2006). *ryoji ikeda | datamatics*. [online] Ryojiikeda.com. Available at: http://www.ryojiikeda.com/project/datamatics/ [Accessed 7 Dec. 2018].

Jaimes, A. and Sebe, N. (2007). Multimodal human–computer interaction: A survey. *Computer Vision and Image Understanding*, 108(1-2), pp.116-134.

Kiefer, C. and Chevalier, C. (2018). What does Augmented Reality Mean as a Medium of Expression for Computational Artists?.

Kramer, G. (1994). *Auditory Display*. Reading, Mass.: Addison-Wesley.

Ryan, N. and Le Couteur, C. (2018). *Adrift: The Secret World of Space Junk*. [online] Adrift: The Secret World of Space Junk. Available at: http://www.projectadrift.co.uk [Accessed 7 Dec. 2018].

van Krevelen, R. (2010). A Survey of Augmented Reality Technologies, Applications and Limitations. *The International Journal of Virtual Reality*, 9(2), pp.1-20.