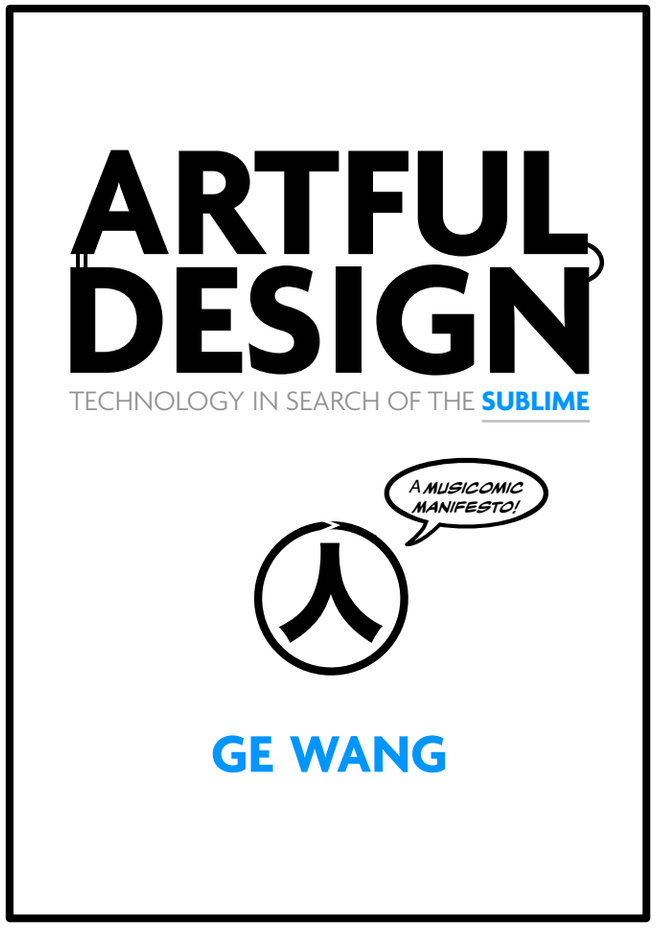
“ChucK: A Strongly-Timed Music Programming Language“

excerpt (pp. 170-175) from *Artful Design*,
Chapter 4 "Programability and Sound Design"



https://**artful.design**/

CHUCK
A STRONGLY-TIMED MUSIC PROGRAMMING LANGUAGE!

CHUCK IS A ***PROGRAMMING LANGUAGE*** FOR ***SOUND GENERATION*** AND ***MUSIC CREATION.*** IT WAS DESIGNED AS A ***TOOL*** FOR RESEARCHERS, COMPOSERS, AND SONIC TINKERERS TO PROGRAM MUSICAL SOUNDS BY WORKING DIRECTLY WITH A NOTION OF ***TIME*** ITSELF. IT IS ***OPEN-SOURCE*** AND FREELY AVAILABLE. (AND, AS I LIKE TO SAY, IT CRASHES EQUALLY WELL ON ALL COMMODITY OPERATING SYSTEMS!) IT HAS A ***PERSONALITY***, AND IS PRETTY EASY TO LEARN.

I STARTED DESIGNING ***CHUCK*** BACK IN 2002 (WHEN I WAS IN GRAD SCHOOL). SINCE THAT TIME, ***CHUCK*** HAS BEEN USED TO CRAFT INSTRUMENTS FOR ***LAPTOP ORCHESTRAS*** AND IS THE AUDIO ENGINE IN ***OCARINA***, RUNNING INSIDE MILLIONS OF PHONES...

WE CAN ***DISSECT*** OUR BLEEP/BLOOP CODE EXAMPLE, WRITTEN IN ***CHUCK***, AND POINT OUT SOME OF ITS ***FUNCTIONALITIES***...

THIS GENERATES A ***SINE WAVE!***

THIS SYMBOL, THE ***CHUCK OPERATOR***, ***CONNECTS*** SOUND-PROCESSING MODULES LIKE A DIGITAL AUDIO ***PATCH CABLE!***

THIS REPRESENTS THE ***AUDIO OUTPUT*** (E.G., LOUDSPEAKERS, HEADPHONES).

```
SinOsc ada => dac;

while( true )
{
    Math.random2f(30,1000) => ada.freq;

    100::ms => now;
}
```

THIS IS A ***LOOP.*** THE CODE WITHIN THE { } IS RUN ***REPEATEDLY!***

THIS GENERATES A ***RANDOM NUMBER*** BETWEEN 30 AND 1000, AND SETS THAT NUMBER AS THE ***FREQUENCY*** OF OUR SINE WAVE.

"NOW" IS ***CHUCK***'S NOTION OF "THE ***CURRENT TIME***": WE TELL ***CHUCK*** WHAT TIME "NOW" ***SHOULD*** BE (E.G., "LET IT BE 100 MILLISECONDS IN THE FUTURE!"), AND ***CHUCK*** MAKES IT SO BY ***WAITING*** THE SPECIFIED DURATION.

MOST COMPUTER LANGUAGES HAVE WAYS TO DEAL WITH TIME (E.G., A "***WAIT***" DIRECTIVE), BUT THESE APPROACHES ARE OFTEN ***COARSE*** AND ***UNPREDICTABLE.*** IN CHUCK, ***TIME*** IS ***ULTRA-PRECISE*** BECAUSE IT IS INFERRED FROM THE ***DIGITAL AUDIO STREAM*** ITSELF. SOUND IN CHUCK IS BOTH THE ***OUTPUT*** AND THE MEANS BY WHICH CHUCK KEEPS TRACK OF ***TIME.***

*THE **FUNCTIONALITIES** OF A PROGRAMMING LANGUAGE DETERMINE WHAT YOU CAN **DO** WITH IT. THE **WAYS** IN WHICH A LANGUAGE **PRESENTS** ITS FUNCTIONALITIES TO YOU CONSTITUTE ITS **AESTHETICS** -- THEY SHAPE HOW YOU **THINK** ABOUT WHAT YOU WANT TO DO.*

```
// create a SinOsc
// called ada
SinOsc ada => dac;

// execute loop?
while( true )
{
    // generate random number as frequency
    Math.random2f(30,1000) => ada.freq;
    // advance time
    100::ms => now;
}
```

CODE

MENTAL FLOW CHART

THE SAME GENERAL FUNCTIONALITY MAY BE AVAILABLE IN DIFFERENT PROGRAMMING LANGUAGES. BUT THE ***SPECIFIC*** WAY A PARTICULAR LANGUAGE EXPRESSES THAT FUNCTIONALITY HAS TO DO WITH THE ***AESTHETICS*** OF THAT LANGUAGE -- AND CHANGES ***HOW*** YOU THINK ABOUT THE TASK AT HAND!

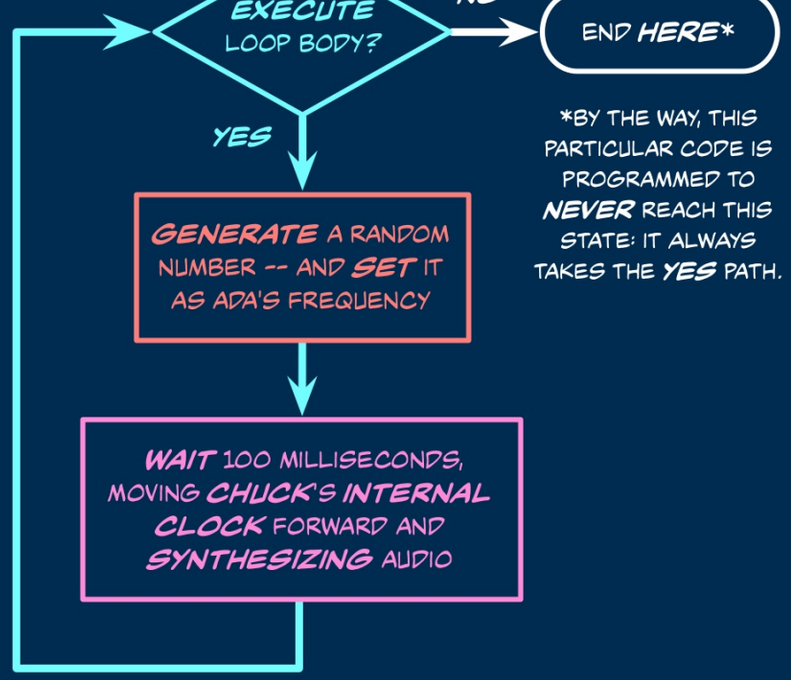
I THINK THIS MAY BE WHY WE HAVE SO ***MANY*** PROGRAMMING LANGUAGES! IT'S NOT SO MUCH THAT THEY ALL ***DO*** DIFFERENT THINGS, BUT THAT EACH ONE MAKES YOU ***THINK*** DIFFERENTLY...

FOR EXAMPLE, THE WAY YOU WORK WITH ***TIME*** IN ***CHUCK*** LENDS ITSELF TO ***REASONING CLEARLY*** ABOUT ***WHEN*** THINGS HAPPEN -- IT'S STRAIGHTFORWARD TO BUILD A ***MENTAL FLOW CHART*** OF WHAT HAPPENS AND WHEN...

START ***HERE***

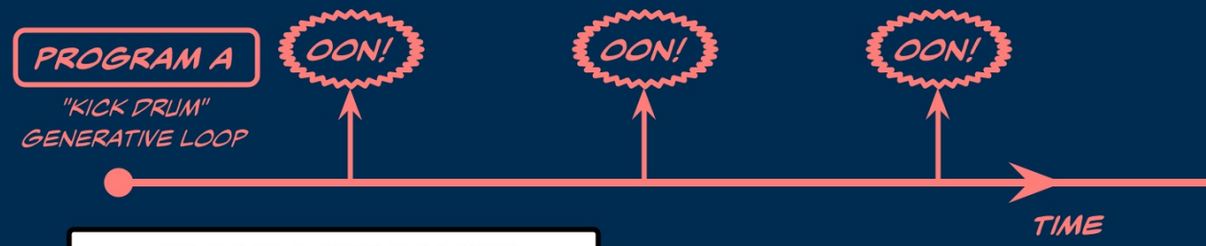
CREATE A SINE WAVE GENERATOR CALLED "ADA"; ***CONNECT*** IT TO "DAC"

THIS IS AN ***INFINITE LOOP*** (THAT MAKES SOUND)!



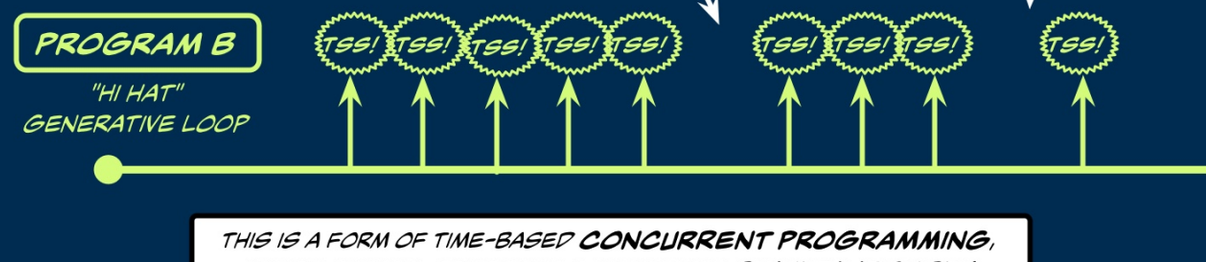
*BY THE WAY, THIS PARTICULAR CODE IS PROGRAMMED TO ***NEVER*** REACH THIS STATE: IT ALWAYS TAKES THE ***YES*** PATH.

BUILDING ON THIS WAY OF THINKING ABOUT ***TIME***, ***CHUCK*** ADDS ONE MORE DIMENSION: THE ABILITY TO RUN MULTIPLE PROGRAMS ***IN PARALLEL***, EACH MANAGING TIME IN ITS OWN WAY. CHUCK USES THIS TIME INFORMATION TO ***AUTOMATICALLY*** AND PRECISELY ***SYNCHRONIZE*** THESE PROGRAMS.

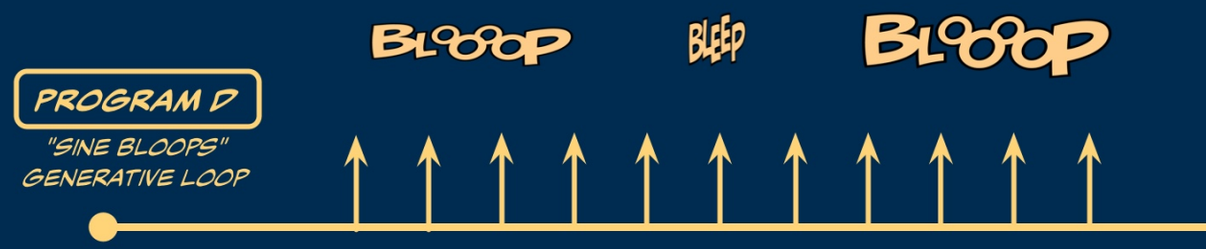


FOR EXAMPLE, HERE IS A ***DRUM MACHINE***, WITH FOUR ***INDEPENDENT*** PROGRAMS OF SOUND-GENERATING CODE, EACH "DOING ITS OWN THING" IN TIME.

WE CAN RANDOMIZE THE AMOUNT OF TIME A PROGRAM WAITS!



THIS IS A FORM OF TIME-BASED ***CONCURRENT PROGRAMMING***, WHERE SEVERAL PROGRAMS CAN EXECUTE ***SIMULTANEOUSLY*** (INSTEAD OF ONE AFTER ANOTHER), SYNCHRONIZED BY TIME.

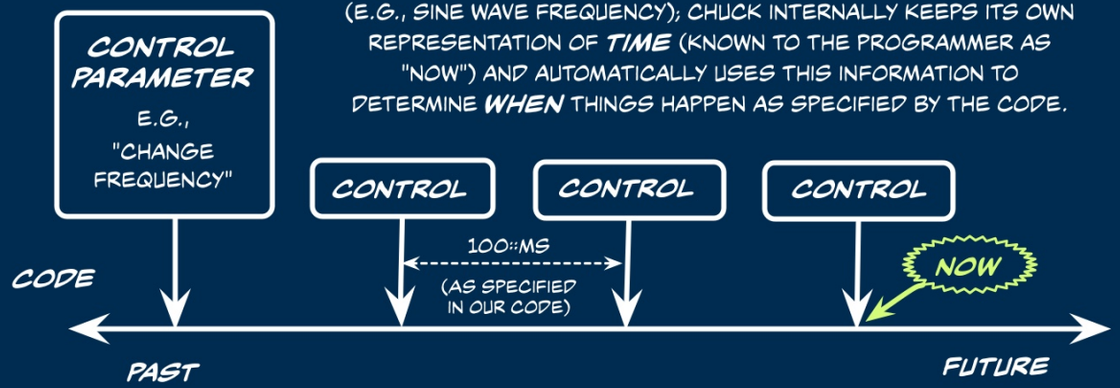


IN ESSENCE, ***TIME*** AND ***CONCURRENCY*** IN ***CHUCK*** ARE TWO SEPARATE DIMENSIONS THAT CAN ***WORK TOGETHER*** -- AS A GENERALIZED WAY TO MODEL ***SOUND*** BOTH AS A ***PHENOMENON OVER TIME*** AND AS A ***MIXTURE*** OF MANY ELEMENTS HAPPENING AT THE ***SAME TIME.***

CHUCK CODE, BY DESIGN, IS A COMPLETE SPECIFICATION OF NOT ONLY ***WHAT***, BUT ALSO ***WHEN*** THINGS HAPPEN. IT ***COMPELS*** THE PROGRAMMER TO BE EVER ***AWARE*** OF TIME WHEN ***WRITING*** CODE, AND TO BE ABLE TO ***REASON*** ABOUT TIME PRECISELY WHEN ***READING*** IT. THIS ***WAY OF WORKING*** AIMS TO FOCUS THE PROGRAMMER'S MENTAL EFFORTS ON SOUND AND HOW IT CHANGES OVER TIME, LEAVING THE LOW-LEVEL DETAILS FOR ***CHUCK*** TO HANDLE.

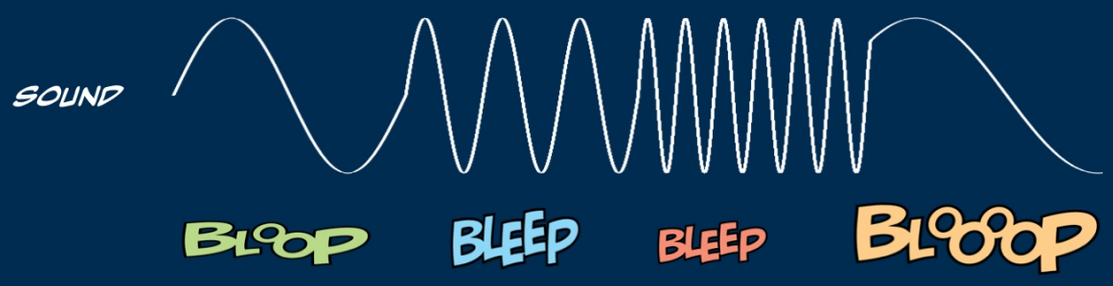
*WHEN YOU RUN THE CODE, **CHUCK** GOES TO WORK! HERE'S A VISUALIZATION OF HOW OUR TIME-BASED CODE BECOMES SOUND...*

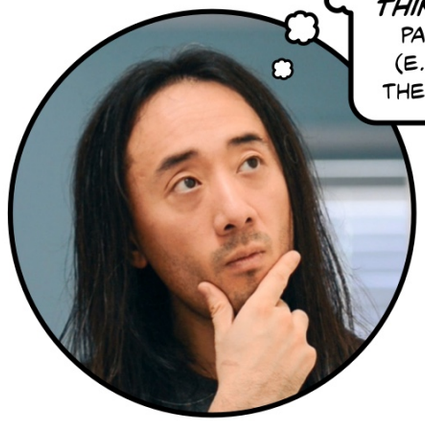
OUR ***CHUCK*** CODE RUNS AT SPECIFIC ***POINTS IN TIME*** (E.G., EVERY 100::MS), CONTROLLING SOUND ***PARAMETERS*** (E.G., SINE WAVE FREQUENCY); CHUCK INTERNALLY KEEPS ITS OWN REPRESENTATION OF ***TIME*** (KNOWN TO THE PROGRAMMER AS "NOW") AND AUTOMATICALLY USES THIS INFORMATION TO DETERMINE ***WHEN*** THINGS HAPPEN AS SPECIFIED BY THE CODE.



DIGITAL AUDIO SAMPLES ARE ***GENERATED*** BY ***CHUCK*** (USING AUDIO GENERATORS LIKE SINOSC) AS ***CONTROLLED*** BY THE CODE WE WRITE (NOTE HOW THE SINE WAVE FREQUENCY CHANGES WITH EACH CODE BLOCK!)

DIGITAL AUDIO SAMPLES ARE NOTHING MORE THAN A ***SERIES*** OF ***NUMBERS*** THAT REPRESENT ***SOUND*** AND CAN BE CONVERTED INTO AN ANALOG SIGNAL THAT WE CAN ***HEAR.***



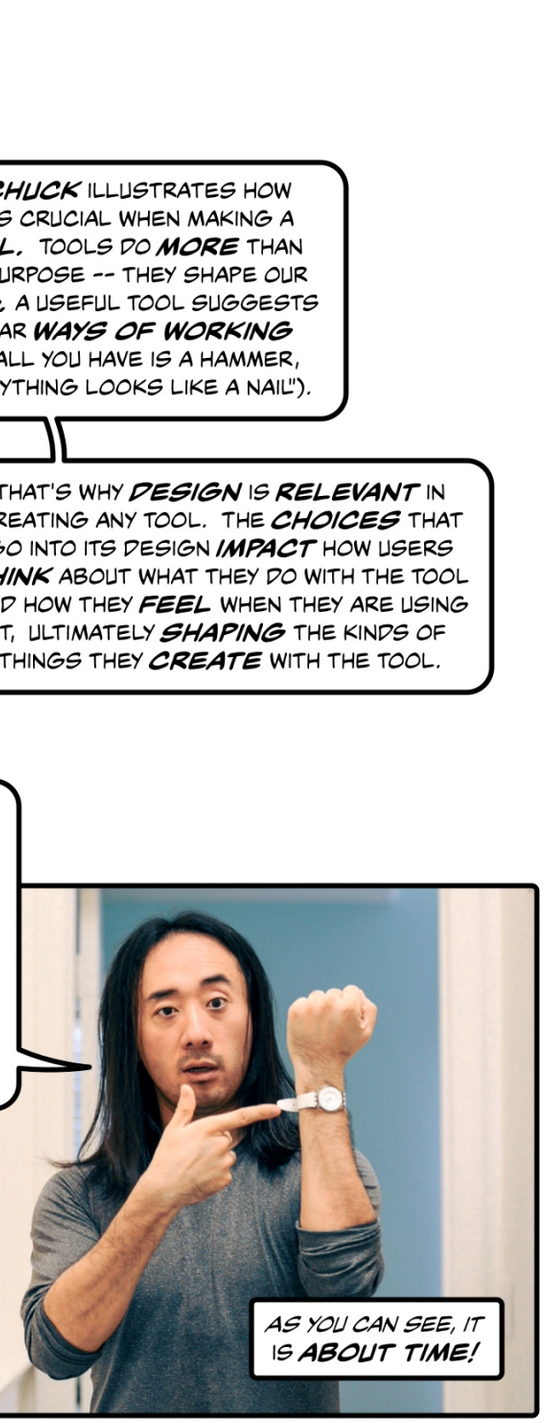


TIME IN ***CHUCK*** ILLUSTRATES HOW DESIGN IS CRUCIAL WHEN MAKING A NEW ***TOOL.*** TOOLS DO ***MORE*** THAN SERVE A PURPOSE -- THEY SHAPE OUR ***THINKING.*** A USEFUL TOOL SUGGESTS PARTICULAR ***WAYS OF WORKING*** (E.G., "IF ALL YOU HAVE IS A HAMMER, THEN EVERYTHING LOOKS LIKE A NAIL").

THAT'S WHY ***DESIGN*** IS ***RELEVANT*** IN CREATING ANY TOOL. THE ***CHOICES*** THAT GO INTO ITS DESIGN ***IMPACT*** HOW USERS ***THINK*** ABOUT WHAT THEY DO WITH THE TOOL AND HOW THEY ***FEEL*** WHEN THEY ARE USING IT, ULTIMATELY ***SHAPING*** THE KINDS OF THINGS THEY ***CREATE*** WITH THE TOOL.

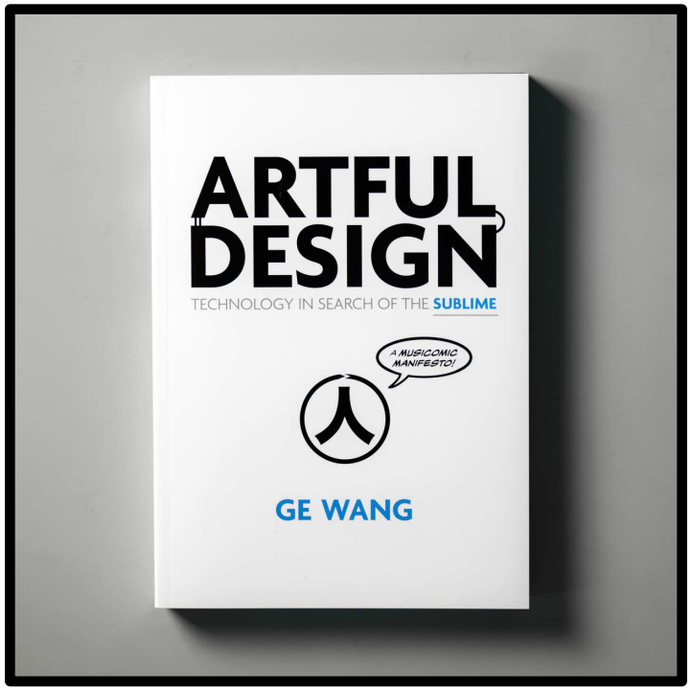
MY ***INTENTION*** IN DESIGNING ***CHUCK*** WAS TO DEVELOP A ***SIMPLE*** YET ***FLEXIBLE*** WAY TO PROGRAM ***TIME*** AS A MUSICAL CONSTRUCT. IT WAS AN EFFORT TO ***HIDE*** THE LOW-LEVEL COMPLEXITIES OF DIGITAL AUDIO SYNTHESIS WHILE ***EXPOSING*** AN ULTRA-PRECISE HIGH-LEVEL WAY TO ***CONTROL*** IT. THE RESULT IS BOTH A ***TOOL*** AND A ***WAY OF THINKING*** TO WORK WITH MUSICAL SOUND, CENTERED AROUND THE FLOW OF TIME.

...I CALL IT ***STRONGLY-TIMED*** PROGRAMMING!



*AS YOU CAN SEE, IT IS **ABOUT TIME!***

WITH THIS IN MIND, LET'S EXAMINE A MORE COMPLEX CASE STUDY OF CONTROLLING SOUND OVER TIME...



https://**artful.design**/