# PD SYNTH

## Bevezető

A PD-SYNTH egy frekvencia modulációs (FM) szintetizátor, mely a Beads java könyvtárat használja fel hangok képzésére. A program a Programozás alapjai 3. tárgy keretein belül készült mint házi feladat.

A frekvencia modulációról rengeteg cikk található az interneten, itt röviden ismertetve lesznek az eljárás alapjai, ami szükséges a program megértéséhez és fejlesztéséhez.

## Frekvencia Moduláció[[1]](#footnote-1)[[2]](#footnote-2)

A szintetizátor alapja, hogy bizonyos frekvenciájú hangokat gerjesztünk. A hangnak a frekvenciája a hordozó frekvenciának hívják. A modulátorok ennek a hangnak a frekvenciáját módosítja így gerjesztve újabb hatású hangokat. Ennek az eljárásnak a matematikájába nem mennék bele, és nem is kell megérteni a program megértéséhez, hiszen a Beads ezeket kiszámolja, nekünk csak a hangokat kell előállítani és a modulátorokat.

Egy modulátornak lehet állítani a frekvenciáját és intenzitását. A frekvenciája torzítja a hordozó frekvenciáját egy bizonyos megadott értékkel, az intenzitás ennek a modulációnak tulajdonképpen az amplitudóját/erősségét állítja.

A PD SYNTH-ben négy modulátor van. A modulátorokat nem csak a hordozó frekvenciára, tehát az alaphangra lehet rákötni, hanem egymást is gerjeszthetik. Tehát a modulátorokat több féleképpen lehet összekötni, sőt, egy modulátor önmagát is gerjesztheti. A modulátorok összekötésének a sorrendjét hívják algoritmusoknak. Rengeteg féle algoritmus létezik, ezekből kettőt implementáltam idő szűkében, ezekről később lesz szó.

## User Manual/Felhasználói útmutató

### Use case

|  |  |
| --- | --- |
| Cím | Billentyűzetek leütése |
| Leírás | A felhasználó irányítja a szintetizátort a billentyűzettel, vagy a grafikus felülettel |
| Aktorok | felhasználó |
| Főforgatókönyv | A felhasználó irányítja a szintetizátort a billentyűzettel, vagy a grafikus felülettel |

|  |  |
| --- | --- |
| Cím | fm-modulátor állítása |
| Leírás | A felhasználó állítja a valamennyi fm modulátort |
| Aktorok | felhasználó |
| Főforgatókönyv | A felhasználó állítja a valamennyi fm modulátort |

|  |  |
| --- | --- |
| Cím | hangkeltő rendszer szabályozása |
| Leírás | A megannyi kontroller egy core hangszabályozót állít, ami szabályozza a kimenő hangot |
| Főforgatókönyv | A kontrollerek állítják a core beállítást |
| Alternatív forgatókönyv | Billentyűleütésre lejátssza a hangot a kimeneti eszközön |

|  |  |
| --- | --- |
| Cím | master szabályozása |
| Leírás | A masteren belül lehet állítani a Gain-t a hangszínt és az egyetlen envelope-ot |
| Aktorok | felhasználó |
| Főforgatókönyv | A masteren belül lehet állítani a Gain-t a hangszínt és az egyetlen envelope-ot |

|  |  |
| --- | --- |
| Cím | beállítások mentése/betöltése |
| Leírás | A felhasználó a beállításokat menti/betölti |
| Aktorok | felhasználó |
| Főforgatókönyv | A felhasználó menti a beállításokat |
| Alternatív forgatókönyv | A felhasználó betölt egy beállítást |

### How to

A programot elindítva egyből az interface tölt be. Az interface-nek három fő egysége van:

Felül találhatók a fő beállítások: itt beállíthatjuk, mely hangkimeneten szeretnénk lejátszani a szintetizált hangot. Ez az első dolog amit be kell állítani, vagy legalább ellenőrizni, hogy valós-e a kimenet ami alapból ki van választva. Miután kiválasztottuk a kimenetet, esetenként várni kell, míg a JVM valóban betölti a kiválasztott hangkimenetet.

Három vízszintes slider-t láthatunk egymás alatt felcímkézve: Az első a gain, ezzel állíthatjuk a gerjesztését a hangnak. Érdemes ezt egy alacsonyabb értékre állítani amennyiben egyszerre sok hangot szeretnénk leütni. A gain alatt található attack és sustain az envelope-ja a hangnak. Az attack-al beállíthatjuk milyen gyorsan (milliszekundumokban mérve) éri el a leütött hang a maximális hangerejét. A sustain pedig, hogy miután elérte a hang a maximális hangerejét mennyi idő után halkuljon el.

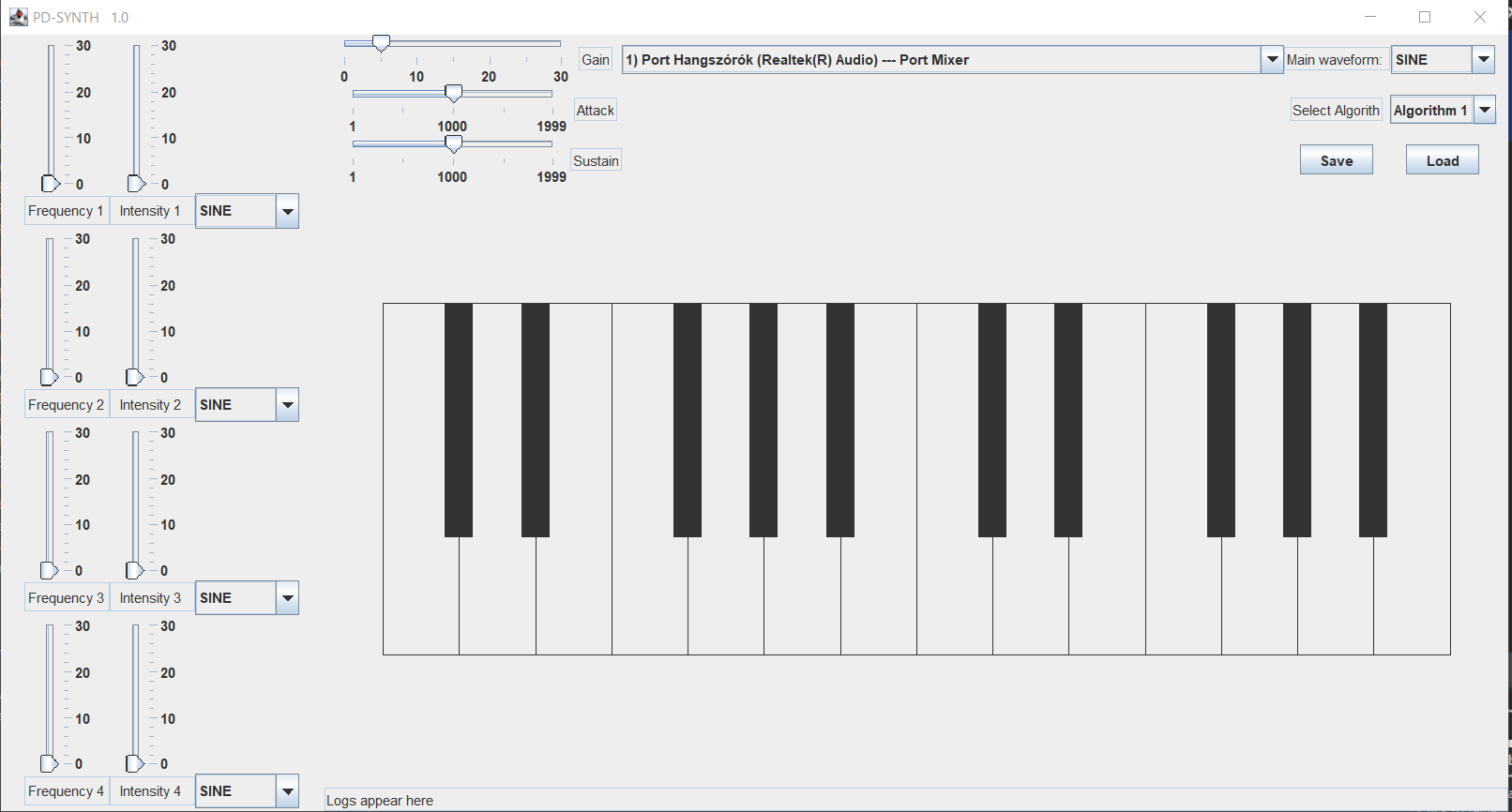
Jobb oldalt két legördülő menü található egymás alatt, az egyikkel a hordozó hang hullámának az alakját állíthatjuk be. 4 lehetőség közül választhatunk: lehet szinuszos a hullám (SINE), fűrész alakú (SAW), háromszög alakú (TRIANGLE), és négyzetes (SQUARE). Az ez alatt található legördülő menüből választhatjuk ki melyik algoritmus szerint szeretnénk összeköteni a modulátorokat. Az Algoritmusokat később ismertetem az Algorithm részben.

Ezek alatt található a Save gomb, ezzel a beállításainkat lementhetjük egy fájlba, és a load gombbal ezt visszatölthetjük. A fájl helyét egy standard fájl böngészővel tehetjük meg.

Középen található a zongora, melyet a jelenlegi verzióval az egerünkkel irányíthatunk. Csupán rá kell kattintani melyik hangot akarjuk lejátszani és a hang lejátszásra kerül.

Bal oldalt található a négy modulátor, amivel szabályozhatjuk a modulátorok beállításait. Mind a négy modulátornak be lehet állítani a frekvenciáját az intenzitását, és hogy milyen alakú legyen a modulátor hullámainak a formája.

Legalul található egy log konzol, amiben minden változtatást és eseményt láthatunk logolva.



## Beads[[3]](#footnote-3)

A program a Beads könyvtárnak csak egy szűk halmazát alkalmazza. Ezek a következők:

### AudioContext

Ez a lelke az egész lejátszónak. Ebből egy generálódik, amikor kiválasztjuk a megfelelő kimeneti eszközt.

### WavePlayer

Ezzel az osztállyal lehet hanghullámokat létrehozni. A frekvencia modulátor is egy ilyen típusú objektum, attribútumai általában egy hullámfrekvencia, a hullámnak az alakja.

### Function

On the fly kell definiálni, ebben különböző wavePlayer-eknek az összeadását/szorzását lehet végrehajtani.

### Envelope

Ez az objektum ha odaadjuk másik Beads objektumnak, képes a paraméterét fokozatosan megváltozni egy bizonyos idő alatt. Ezt az időt milliszekundumokban kell megadni. Például egy WavePlayernek meg lehet változtatni a frekvenciáját vele.

### Gain

Ezzel lehet a gerjesztését állítani a kimenetnek. A PD SYNTH-ben csak ennek a komponensnek van envelope-ja.

## Struktúra

A PD SYNTH rendszere két részre/package-re van bontva: Az egyik package felelős a grafikus megjelenítésért (GUI) a másik a háttérszámolásokat végzi (SynthSystem). Ezen kívül még két package van, az egyikben a tesztek találhatóak, a másikban az az inicializáló osztály, amiben van a main(Application).

## SynthSystem

### AuidioIOSelector

Ez az osztály nyeri ki a rendszernek elérhető hang be és kimeneti eszközeit. Fő metódusa a getAudioOutputs mely egy String tömbbel tér vissza, amiben az egyes kimeneti és bemeneti eszközök találhatóak. Ezeknek az adatoknak a kigyűjtését segíte a JavaSoundAudioIO. Sajnos javában nem lehet elkülöníteni a hang kimeneti és bemeneti eszközöket. Egyéb kellemetlenség, hogy az előbb említett osztálynak nincs olyan metódusa, amely visszatér olyan String-el ami tartalmazza a keresett eszközöket, hanem csak olyan, ami Standard kimenetre írja ki. Ezért egy trükkös módszerrel, ezt begyűjtöm egy String-be. Sajnos nagyon sok extra információt is kigyűjt az eszközökről, ennek a szűréséről a removeAudioDescriptionFromOutputLog privát metódus felelős. Ez az osztály hozza létre az AudioContex-t annak alapján, hogy milyen kimeneti eszközt választunk. Ezért a selectMixer metódus felel.

### DataOperator

Ez az osztály gyűjti ki a szükséges információkat az Interface-től. Ez egy szerializálható osztály, a mentés miatt létezik. Egyik fő metódusa a getData, amit ha meghívunk, begyűjti az Interface-en tárolt valamennyi adatot, és feltölti vele az adattagjait. Másik metódusa a setData, aminek a meghívásával az Interface valamennyi adattagját módosítja arra az értékre, ami a saját adattagjai hordoznak.

### Keys

Enumerációs osztály, mely tartalmazza minden hang enumerációját. Azért hoztam létre, hogy a SynthEngine könnyen el tudja kérni a legenerálandó hangokat a Piano-tól, illetve ezek alapján a Notes osztálynak is könnyen megmondhatjuk milyen hangoknak a frekvenciáira vagyunk kíváncsiak. Az enumokat lehet kasztolni integer változóvá, a C értékétől kezdve számolva. Ez a Notes osztály számára hasznos tulajdonság.

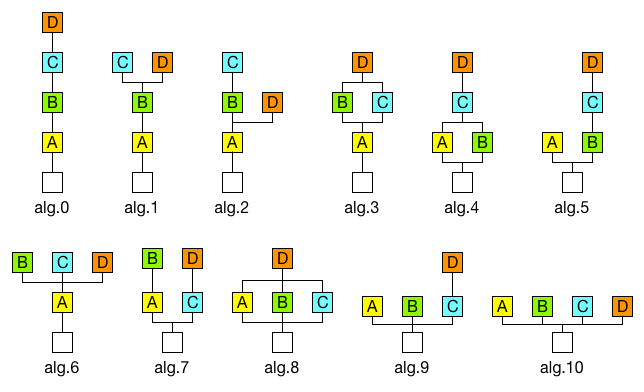
### Notes

Ez az osztály lényege, hogy egy adott Key enumból visszatérjen az adott hang frekvenciájával. Az eredeti terv az volt, hogy az alaphang frekvenciája változtatható legyen, ám ennek implementálására nem jutott idő. Egy egyszerű fizikai egyenletből meg lehet kapni az összes hang frekvenciáját[[4]](#footnote-4). Privát adattagja az A hangnak az alapfrekvenciája (440Hz). Ebből a calculateCFromBaseNote kiszámolja a C hang frekvenciáját. Ez azért hasznos, hisz így a getFrequencyFromPressedkey-nek az argumentumának adott enumerációjának az integerré való kasztolása (getValue) megadja a távolságot (hangokban mérve) a C hangtól, és így könnyebb számolni a gerjesztett hangok frekvenciáját.

### SynthEngine

A szintetizátor agya. Összegyűjti a hang szintetizálásához szükséges információkat az Interface-től és azokból legenerálja a hangot. A lenyomott hangok egy tömbben vannak tárolva. Ennek az oka az, hogy eredetileg billetyű támogatást is terveztem a programba, ám idő hiányában, ezt nem sikerült implementálni.[[5]](#footnote-5) A hang generálásáért a startPlaying metódus felel, melyet a Piano hív meg, amikor leütnek egy billentyűt. Ekkor létrejönnek azok a Beads elemek, amik szükségesek a hang generálásához. Minden elem létrehozásához, az interface egyes komponenseit kérdezi le, hogy mik a beállítások. Létrejön a kezdeti envelope (startDelay/attack) és a vég envelope (endDelay/decay), beállítja a hang erősségét, tehát létrejön egy Gain. A szükséges WavePlayer-t a beállított algoritmus producedWavePlayer metódusa adja meg. Ebből szólal meg a hang az ac.start parancs által.

## Algorithm[[6]](#footnote-6)

Az Algorithm interfész segítségével könnyen tudunk implementálni algoritmusokat. Minden algoritmusnak kell, hogy legyen egy neve, és egy producedWavePlayer metódusa. Az utóbbi hozza létre az adatokból a végleges Waveplayert amit majd az AudioContext lejátszik. Én idő szűkében két algoritmust implementáltam, de nagyon egyszerű az algoritmus halmazt bővíteni.

1. ábra: 4 modulátor összekapcsolási lehetőségei. Ebből a 0-ás és 7-es lett implementálva. Forrás:https://github.com/mohayonao/fm-synth

### Default Parallel

Az első ábrán látható 7.-ik algoritmust valósítja meg. Ebben az A és C a fő modulátorok a programban az 1-es és 2-es modulátoroknak felelnek meg. Ezeket gerjeszti B és D modulátor, az 1-es modulátort a 3-as gerjeszti, a 2-es modulátort pedig a négyes.

### DefaultSerial

Az első ábra 0-ás algoritmusát valósítja meg. A fő modulátor ami a hordozó frekvenciát modulálja az 1-es modulátor, és sorban követi egymást a többi.

## GUI

### Interface osztály

A GUI-t az Interface osztály hozza létre és kezeli. Minden Swing elem tárolva van benne, és ő is inicializálja. Ezek az elemek mind publikusak, mert a backend rendszer az interface-en keresztül éri el az adott komponensek beállításainak az értékét. A layout-ja alapvetően egy BorderLayout melynek az Északi(NORTH) Déli (SOUTH) Nyugati (WEST) és Közepe (CENTER) része van használva. Az északi részben található a hordozó frekvencia állapotát beállító elemek(EnvelopeSetter, VolumeSetter, MainWaveFormSetter), a kimeneti eszközt beállító ComboBox (AuidiiIOSelectorElement), az algoritmus állító gördülőmenü (AlgorithmSetter) és a mentés és betöltés gombok (FileManager). A Nyugati oldalon találhatók a modulátorok tulajdonságait kezelő komponensek. Középen található a zongora (Piano), amit az egerünkkel kattintva állíthatjuk elő a kívánt hangot. Délen egy szimpla logger található, ami mindent kiír, amit módosítunk, illetve végzünk a programban.

Minden egyes komponens egy osztályként van realizálva. Minden osztály tartalmazza a komponens(eit) és az adatait, ami be lett állítva rajta. Minden komponenst az Interface inicializál és helyen el önmagán. Minden komponens megkapja az Interface-t paraméterként, hogy azon tudjon loggolni.

## GUI komponensek listája:

Mindegyik grafikus komponenst egy osztályként realizáltam, mely körülbelül illeszkedik egy olyan sémára, hogy minden grafikus osztály egy vagy több hasonló elemet tárol, Swing komponense privát adattag, és az hogy mit választottak a Swing komponensben szintén privát adattagként van tárolva.

|  |  |  |
| --- | --- | --- |
| Komponens osztály neve | Rövid leírás | Megvalósítás |
| AlgorithmSetter | Egy legördülőmenü, melyen beállíthatjuk mely algoritmus szerint szeretnénk összekapcsolni a modulátorokat. | Adattagjai a swing komponens, egy egyszerű tömb mely tárolja az algoritmusok nevét, egy változó mely tárolja az adott algoritmust, és egy egész mely a különböző algoritmusok számát tartja nyilván. |
| AudioIOSelectorElement | Egy legördülő menü, melyen kiválaszthatjuk gépünk melyik hangkimenetén szólaltatjuk meg a szintetizátort. | Adattagjai a swing komponens. A kimenet kiválasztása egy olyan bonyolult feladatnak bizonyult, mely felvetette igényét, hogy egy külön osztály valósítsa meg a kimenetek kezelését, ez a AudioIOSelector osztály. Ennek az osztály selectAuidoInput metódusát meghívva választja ki a megfelelő kimentet. |
| EnvelopeSetter | Egy slider, melyen állíthatjuk a hang envelope-ját, vagyis az ívét tehát milyen sebességgel érje el csúcs erejét a hangját (Attack), és végül milyen sebességgel halkuljon el (Decay). | Tárolja mind az Attack mind a Decay swing komponenst és értékét. |
| FileManager | Ez az osztály felelős a beállítások mentéséért és beolvasásáért. | Egy segédosztályt alkalmaz, a DataOperator-t amely az interface adatait gyűjti, amely szerializációval kiír egy file-ba. A mentés és betöltés helyét egy JFileChooser-el lehet kiválasztani. Továbbá két JButton-t is tartalmaz, amivel elindítható a mentés és betöltés. |
| FMElement | Mindegyik modulátort tartalmazó osztály. Mind a négy modulátornak van két slider-e amivel a frekvenciát és intenzitását lehet állítani, és egy legördülő menüje, amivel a modulátor hangalakját lehet kiválasztani. | A slider-ek és gombok és adatok is egy-egy négy elemű primitív tömbben vannak tárolva. GridBagLayout-al mindegyik elemet rendeztem egy Panel-be, hogy az interface-ben már csak nyugatra beszúrni kelljen. |
| Logger | Egy szimpla TextBox amire loggolhatnak a komponensek az interface-en keresztül | Trivális |
| MainWaveFormSetter | Ezzel állíthatjuk be a a hordozó hang alakját. | Tartalmaz egy ComboBox-ot melyen ezek a formák kiválaszthatók. A Buffer típusú adattag tárolja melyik lett kiválasztva. Mivel a Buffer osztálytól nem lehet elkérni a nevét, egy String változóban tárolja, hogy mi a kiválasztott hangalak neve. Ez a szerializáció miatt van így. |
| Piano | Ezzel az osztállyal később részletesen lesz leírás | |
| VolumeSetter | A szinti gerjesztését állító slider-t tartalmazza. | Tartalazza a slider-t és értékét |

## Piano osztály

A piano osztály alacsony szintű grafikával rajzolja ki a főpanel közepén látható zongorát. Ezen kívül gyűjti mely hangok lettek lenyomva az egér által, és szól a SynthEngine-ek, hogyha hangot kell gerjesztenie.

A paint-t metódusában fut le az algoritmus, mely a zongorát rajzolja, ezt két for ciklussal oldottam meg, az egyik a fehér a másik a fekete billentyűket rajzolja le. A fekete billentyűket minden 3.-ik és 7.-ek helyen ki kell hagyni, tehát ha az iterátor 7-el adott maradéka 3 vagy 0, akkor nem rajzol fekete billentyűt. Minden billentyű bal sarkának a pozíciója mentve van egy-egy Point primitív tömbben, külön a fekete külön a fehérbillentyűk, ennek az az, oka, hogy tudjuk, hogy hova klikkelünk az egerünkkel.

A fekete billentyűk érzékeléséhez és fehér billentyűk érzékeléséhez külön két-két segéd függvény a whichWhiteKeyHasBeenPressed és a whichBlackKeyHasBeenPressed, ami mindkettő visszatér azzal az index számmal, hogy a fekete billentyűk tömbjében és a fehér billetyűk tömbjének hányadik tagja lett lenyomva, ezekből az értékekből pedig a createKeyValoueOutOfWhiteKey és a createKeyValueOutOfBlackKey Keys típusú enumerációt alkot, amit a SynthEngine már tud értelmezni.[[7]](#footnote-7) Az átalakítás logikája a mousePressed függvényben található. Key release-kor szól a SynthEngine-ek a zongora, hogy abbahagyhatja a hang játszását.

1. Frekvencia modulációs (és egyéb hanggerjesztés) beads-el: <https://evanxmerz.com/soundsynthjava/Sound_Synth_Java.html> [↑](#footnote-ref-1)
2. Frekvencia moduláció alapjai: <https://en.wikipedia.org/wiki/Frequency_modulation_synthesis> [↑](#footnote-ref-2)
3. Beads honlapja: <http://www.beadsproject.net/>

   Beads könyvár részletes dokumentációja: <http://www.beadsproject.net/doc/> [↑](#footnote-ref-3)
4. https://www.translatorscafe.com/unit-converter/en-US/calculator/note-frequency/ [↑](#footnote-ref-4)
5. Néhány része ennek az implementációnak bent maradt a programban, hiszen ezt később magamnak szeretném még implementálni. Ezért van az, hogy a hangok (notes) illetve a lenyomott billentyűk (whichKeysHasBeenPressed) gyűjteményben vannak. [↑](#footnote-ref-5)
6. Videó mely elmagyarázza az algoritmusok működését: <https://www.youtube.com/watch?v=fue5-gS_4l0> [↑](#footnote-ref-6)
7. Ezek alapból privát metódusok, tesztelés miatt lettek publikussá téve. [↑](#footnote-ref-7)