



République Française
Ministère de
L'enseignement Supérieur
et de la Recherche



RAPPORT DE PROJET

Puzzle musical interactif

Présenté à

Université de Caen Normandie

en vue de valider

l'unité projet du Master II IMALANG

Réalisé par

AIT YAHIA Yassine

Tuteur Universitaire : Youssef CHAHIR

Année : 2015/2016

Dédicace

Il m'est particulièrement agréable de profiter de cette occasion, pour rendre un hommage particulièrement sincère à travers cet ouvrage, à tous ceux qui me sont chers, à tous ceux qui m'ont soutenu moralement et matériellement.

Je dédie donc ce travail :

A mes très chers et honorables parents, mes grands-parents, mes oncles, mes sœurs ainsi que tous les membres de ma famille qui m'ont toujours encouragé, aidé, guidé vers le bon chemin, par leur bonté, leur tendresse, leur générosité, leur éducation et leurs riches et indélébiles conseils.

A mes chers amis, les collègues ainsi que tous les enseignants et les responsables.

A mon encadrant M. Youssef CHAHIR et tous les professeurs de L'UNICAEN.

Et à toute personne qui a contribué à la réussite de ce projet que ce soit de près ou de loin.

Qu'ils trouvent à travers ce travail ma sincère reconnaissance.

Remerciement

Je tiens à remercier toutes les personnes qui ont participées de différentes façons à la réussite de mon projet et plus particulièrement les personnes que je cite ci-dessous.

M. Youssef CHAHIR, Responsable du Master II IMALANG à l'UNICAEN pour toutes ses informations, sa disponibilité et son aide.

Enfin, j'espère bien que ce travail fera la fierté de toute personne qui a participé directement ou indirectement à sa réalisation.

Sommaire

Dédicace	1
Remerciement.....	1
Sommaire	2
Tables des Figures	3
I. Introduction	4
II. Gestion de projet	5
1. Objectif.....	5
2. Définition des besoins	5
2.1. Quelques jeux proposés	6
3. Analyse des besoins.....	8
4. Déroulement du projet.....	12
5. Planification du projet	12
III. Description de la solution proposée	14
1. <i>Reactivision</i>	14
1.1 Présentation de la solution	14
1.2 Architecture.....	14
1.3 Analyse	20
IV. Développement des clients <i>PureData</i>	22
1. Jeu de classification.....	22
1.1 Arbre de décision.....	22
1.2 Programme <i>PureData</i>	23
2. Jeu de placement des cartes selon leurs positions	24
2.1 Arbre de décision.....	24
2.2 Programme <i>PureData</i>	25
V. Conclusion.....	26
Abréviations	27
Annexe	28
Références.....	29

Tables des Figures

Figure 1: Classification	6
Figure 2 : Structure des pièces.....	7
Figure3 : Positionnement des pièces	7
Figure 4 : Composition des pièces.....	7
Figure 5 : Tableau descriptif des différentes technologies	11
Figure 6 : Planning du projet détaillé Final	13
Figure 7 : Planning du projet détaillé Final	14
Figure 8 : Exemple de la <i>Reactable</i>	15
Figure 9 : Type de caméra utilisée	16
Figure 10 : Exemple de marqueurs fiduciaux.....	16
Figure 11 : Champs de vision de <i>Reactivision</i> avec deux axes x et y	18
Figure 12 : Recevoir et déballer les informations.....	19
Figure 13 : Exemple de programme « lecture un fichier wav ».....	19
Figure 14 : Exemple pour travailler avec des conditions.	20
Figure 15 : Tableau descriptif des Technologies utilisées.....	21
Figure 16 : Arbre de décision du jeu de classification.	22
Figure 17 : Programme <i>PureData</i> du jeu de classification.....	23
Figure 18 : Jeu de placements des cartes fiduciales.	24
Figure 19 : Programme <i>PureData</i> du jeu de placement des cartes.	25
Figure 20 : Les objets <i>PureData</i>	28

I. Introduction

Aujourd'hui, nous vivons dans un monde qui est, de plus en plus, lié au multimédia et aux différentes possibilités qui peuvent offrir ses supports, tant au niveau de l'interactivité qu'au niveau du design. Le développement des nouvelles technologies est de plus en plus important. C'est dans ce but que je souhaite créer un puzzle musical interactif dans lequel l'utilisateur pourra de façon simple assembler des segments musicaux sous forme de pièces de puzzle.

Ce projet se réalise dans le cadre du module projet du Master II IMALANG à l'UNICAN. Ce dernier me permet de concevoir le déroulement d'un projet tout en analysant un problème complexe.

L'objectif, pour l'utilisateur, est de manipuler des objets afin de répondre aux différentes consignes. En effet, chaque objet ou pièce de puzzle permettra d'effectuer une action telle que lancer un extrait de musique, vérifier le résultat du jeu, etc. Les consignes potentielles seraient de former différents tas constitués de pièces de puzzle, de reconstituer un morceau avec les différents extraits contenus dans chaque pièce de puzzle. La principale contrainte est que l'utilisateur doit placer les pièces dans un endroit plus ou moins précis afin de valider sa composition.

II. Gestion de projet

La gestion de projet est une démarche visant à organiser son bon déroulement depuis la première étape jusqu'à sa mise en œuvre. Dans mon cas, j'ai utilisé le « cycle en V ».

1. Objectif

L'objectif est de construire un morceau musical en assemblant les pièces et en lançant l'écoute. Chaque pièce du jeu devra, donc, permettre aux utilisateurs d'écouter le morceau de musique correspondant.

L'objectif est de parvenir à réaliser deux jeux :

- Premier cas : permet de faire de la « Classification » et réussir à distinguer à base de l'identification deux types de musiques différents par exemple « arabe » et « disco ».
- Second cas : l'utilisateur va former son jeu de telle sorte que l'emplacement des pièces va influencer la réussite ou l'échec du jeu.

Ce projet permettra de donner la possibilité à un utilisateur de lancer un jeu, d'identifier chaque pièce en écoutant le morceau correspondant et aussi de mettre en place des différentes pièces dans le bon ordre et sous une forme précise selon la solution abordée (aligné, cercle, carré, ou dans une échelle musicale...).

2. Définition des besoins

Le but de ce projet est de réaliser un jeu avec des pièces de puzzle. L'utilisateur construit ou reconstitue un morceau en assemblant ces pièces et en lançant l'écoute.

Chaque pièce de puzzle doit être identifiée. L'identification de chaque pièce permet d'écouter l'extrait musical correspondant et repérer sa position. L'utilisateur doit alors réaliser un lot à l'aide de ces pièces et valider sa composition. La validation se fera, soit en cliquant sur un bouton « validation », soit par le biais d'une pièce de puzzle; un signal sonore et/ou lumineux informera l'utilisateur du bon ou mauvais résultat final.

Principalement, l'utilisateur manipulera des pièces de puzzle. Il faut en compter entre 8 et 16 pièces qui seront de forme rectangulaire de préférence. Elles seront de matière plastique ou cartonnée avoisinant les dimensions 6 cm sur 4 cm. Le prix de chaque pièce ne dépasse pas 3 euros.

Les pièces devront visuellement être identiques ou, en tout cas, ne devront pas avoir de similarités visuelles facilitant l'assemblage de ces dernières. Dans ces ateliers l'ouïe devra être le seul sens mis en avant chez les utilisateurs.

2.1. Quelques jeux proposés

a) Classification

L'objectif de la classification est la réalisation de deux ou trois tas avec les pièces sur le support afin de valider les tas formés. Dans cet atelier la précision du positionnement n'est pas très importante, on peut se contenter d'une précision à une quinzaine de centimètres près (~15 cm).

Dans les illustrations suivantes, on admettra qu'il faut réaliser deux tas. Ajouter un tas ne modifiera pas le système.

L'utilisateur identifie chaque pièce en écoutant le morceau et la dispose dans le tas correspondant.

Les tas seront réalisés selon une identification des morceaux. Dans un jeu, il sera demandé à l'utilisateur de distinguer deux morceaux différents qui seront joués par différentes pièces. Dans ce cas-là, un tas sera composé de l'ensemble de pièces qui donne un morceau complet.



Figure 1: Classification

b) Analyse

Placer les pièces dans un ordre précis afin d'avoir une composition valide.

Fonctionnement de l'atelier :

L'utilisateur va former son jeu à base de la structure (À, B, C, D,...), de telle sorte que l'ordre des pièces va influencer la réussite ou l'échec du jeu.

Positionnement des pièces (0,5 à 2 cm)

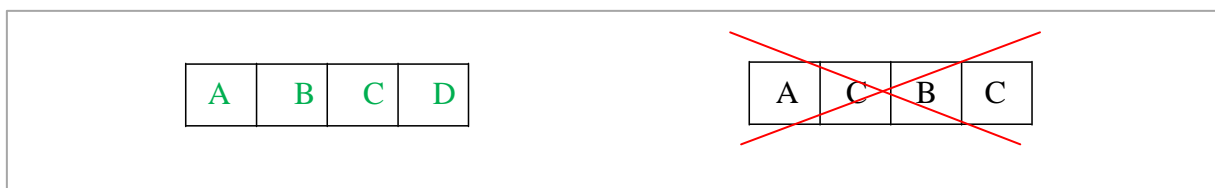


Figure 2 : Structure des pièces

c) Positionnement

L'activité « positionnement » demande à l'utilisateur de placer les pièces à des endroits assez précis. La vérification du positionnement validera, ou non, la mise en place des cartes par l'utilisateur.

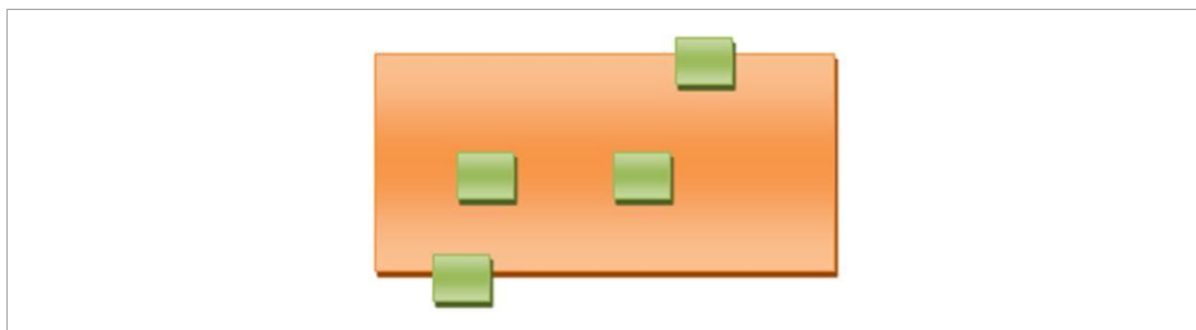


Figure 3 : Positionnement des pièces

d) Composition

Dans cette activité le placement des pièces se réalise de telle sorte que l'implication d'une nouvelle pièce dans la composition modifie la pièce originale.

On a plusieurs motifs (A, B, C, D, S), certains vont ensemble en se superposant (par exemple des S/CS), on compose tout cela.

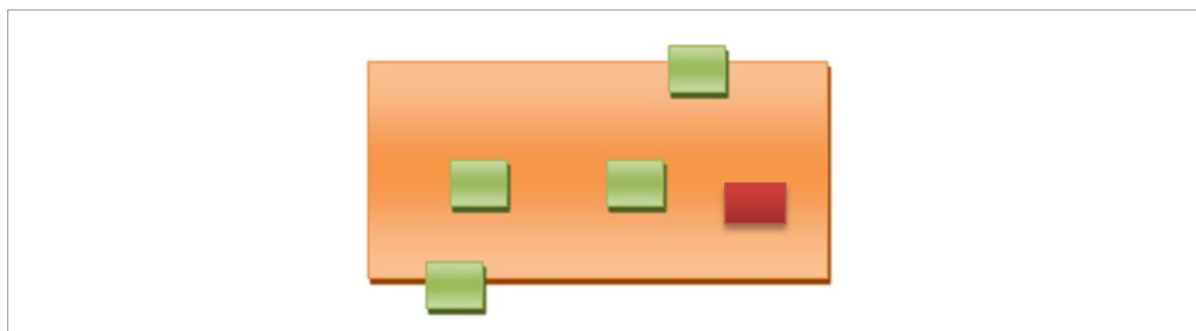


Figure 4 : Composition des pièces

3. Analyse des besoins

Afin de répondre aux besoins définis, j'ai procédé à une étude des différentes technologies pour trouver une solution à la problématique posée :

- *Leap Motion*
- *Kinect*
- *Xbee*
- *ZigBee*
- *Arduino*
- *Propeller parallax*
- *RFID*
- Capteur *piézo* et disque piézoélectrique,
- *Reactable*, *Reactivision* et les marqueurs fiduciaires.

Technologie	Avantages	Inconvénients	Prix
Leap Motion	<p>Un pas dans le futur : nouvelle technologie (suscite la curiosité et souligne le progrès des technologies)</p> <p>Précision : 200 fois plus sensible que le Kinect</p> <p>Compatibilité : Windows et Linux</p>	<p>Trop récent : pas d'application développée, interrogation sur le succès de cette technologie</p> <p>Fatigue physique (enfant et personnes âgées, personnes handicapées, ...)</p> <p>Perte du côté matériel (touché)</p> <p>Difficulté de prise en main</p> <p>Développement d'une application tierce</p>	<p>80 euros le boitier (+ prix des applications futures) + tablette</p>
Kinect	<p>Succès au niveau du public des jeux vidéo</p> <p>Facilité de prise en main</p> <p>Novateur</p>	<p>Nécessite de grands espaces</p> <p>Fragile</p> <p>Mauvaise perception des mouvements (surtout les plus petits)</p> <p>Initialement conçu pour les jeux vidéo</p> <p>Développement d'une application tierce</p>	<p>149 euros + tablettes</p>

<p>Xbee</p>	<p>Fréquence porteuse : 2.4Ghz</p> <p>Portées variées : assez faible pour les xbee 1 et 2 (10 - 100m), grande pour le xbee Pro (1000m)</p> <p>Faible débit : 250kbps</p> <p>Faible consommation : 3.3V @ 50ma</p> <p>Entrées/sorties : 6 10-bit ADC input pins, 8 digital IO pins</p> <p>Sécurité : communication fiable avec une clé de chiffrement de 128-bits</p> <p>Simplicité d'utilisation : communication via le port série</p> <p>Flexibilité du réseau : sa capacité à faire face à un nœud hors service ou à intégrer de nouveaux nœuds rapidement</p> <p>Grand nombre de nœuds dans le réseau : 65000</p> <p>Topologies de réseaux variées : maillé, point à point, point à multipoint</p>	<p>pas un empatement de type « labdec » ; alim 3.3V</p>	<p>Faible coût : ~ 25€</p>
<p>ZigBee</p>	<p>Sa ressource mémoire est de 4 à 32 ko. (le wifi est >1Mo et le Bluetooth >250ko)</p> <p>Il utilise une bande passante de 20 à 250 kbps</p> <p>Couvre une portée entre 1 et 100m</p> <p>Son principal point fort est sa faible consommation, (100 jours à 2 ans pour les piles contre 0.5 à 5 jours seulement pour le wifi par exemple).</p>	<p>Dépendance car obligation d'acheter un module Xbee pour connecter la carte avec l'ordinateur</p>	<p>Faible coût : ~ 25€</p>

<p>Arduino</p>	<p>ARDUINO = 1 carte à microcontrôleur + 1 outil de développement + 1 communauté active Le logiciel et le matériel sont open-source :</p> <ul style="list-style-type: none"> • peu coûteux, • platine de base très compacte, • simple et facile à mettre en œuvre, • large bibliothèque d'exemples, • forums pour des conseils sur le net. 	<p>Peu adapté pour la gestion de plusieurs périphériques en parallèle (contrôle de servos + réception de messages IR ou radio + émission de musique + détection d'obstacles)</p> <p>Limité lorsqu'il faut traiter des signaux très brefs</p>	<p>Prix d'une carte Arduino uno = 20 euros</p> <p>Logiciel = 0 euros</p> <p>Support et assistance = 0 euros (forums)</p>
<p>Propeller parallax</p>	<p>Le <i>microcontrôleur</i> multicœurs (8 cœurs) Propeller, kits, et des conseils de développement</p> <p>Programmable en <i>Spin</i></p> <p>Utilisé dans de nombreuses industries y compris, la fabrication, le contrôle de processus, de la robotique, de l'automobile et des communications</p>	<p>Peu d'utilisateurs en France (début 2008)</p> <p>Quelques importateurs en France (Selectronic, LCie)</p> <p>La programmation en langage <i>Spin</i> est un peu plus complexe que celle en Basic des Stamp, il faut surtout s'habituer à penser « programmes parallèles »</p>	<p>Entre 25 et 75 euros</p>
<p>RFID</p>	<p>Méthode pour mémoriser et récupérer des données à distance en utilisant des marqueurs appelés « radio-étiquettes »</p> <p>Interaction avec des tags RFID à une distance de 5cm</p> <p>Aussi avec des dispositifs NFC (Near Field Communication) tels que des smartphones, des cartes bancaires, des dispositifs de paiement sans contact, etc</p> <p>Module RFID se connecte à Arduino au travers d'un <i>shield XBee</i>.</p> <p>La communication entre l'Arduino et le module RFID se fait par le port série (UART)</p>	<p>Une distance de lecture limitée à 50 cm en France compte tenu des niveaux de puissance autorisés aujourd'hui</p> <p>Une perturbation possible du signal radio par la présence de métal</p> <p>Les ondes radio pourraient être nuisibles pour les consommateurs</p>	<p>Pour les étiquettes actives, le prix est encore élevé, par rapport aux codes à barres</p> <p>Comptez de 10 à 15 euros pour une étiquette RFID active et de 10 à 20 centimes pour étiquette passive</p>

<p>Capteur piézo & Disque piézoélectrique</p>	<p>Utilisé en tant que détecteur de chocs, ce capteur fonctionne en analogique donc perçoit des différences entre les chocs forts et les coups faible. Il permet de choisir la sensibilité du capteur et de l'adapter à diverses situations.</p> <p>Ce capteur fonctionne en analogique donc perçoit des différences entre les chocs forts et les coups faibles</p> <p>Disque piézoélectrique sans boîtier constitué d'un disque métallique sur lequel est collé un élément <i>piézo-céramique</i>.</p> <p>Une tension alternative appliquée sur cet élément provoquera une contraction ou une expansion diamétrale de l'élément, générant un signal sonore</p>	<p>Limitation d'utilisation</p> <p>Dans le cadre de ce projet ce disque sera utilisé comme pièce de puzzle mais le problème est que cette pièce sera fixée par un fil lié au capteur <i>piezo</i> donc pas de portabilité</p> <p>Contrôler par Arduino donc l'obligation de le brancher avec Arduino</p> <p>Il lui faut une bonne liaison mécanique avec le support pour bien percevoir les vibrations</p> <p>Il doit y être fixé solidement</p>	<p>1 Capteur piézoélectrique : 19,50 euros TTC</p> <p>1 Disque piézoélectrique 2,50 € TTC</p> <p>= 22 € TTC</p> <p>Par pièce de puzzle</p>
<p>Reactable, Reactivision et Les marqueurs fiduciaires</p>	<p>Un instrument de musique électro-acoustique pour utilisateurs multiples fabrication simple et manuelle utilisant le dessus d'une table vitrée comme interface utilisateur tangible, plusieurs personnes peuvent partager simultanément le contrôle de l'instrument.</p> <p>Basée sur le déplacement d'objets physiques sur la surface de la table, la <i>Reactable</i> génère également des topologies sonores différentes.</p> <p>Open Source multi-plateformes permettant la captation rapide de marqueurs fiduciaires attachés à des objets physiques. Cette application autonome envoie des messages de contrôle OpenSound par UDP à toute application client suivie. Elle rend effectif le protocole TUIO pour la transmission de l'état d'objets tangibles et pour le contrôle d'un multi-touche sur une surface plane.</p> <p>Conçue pour traquer des marqueurs fiduciaires spécifiques.</p>	<p>Dans le cadre de ce projet, l'utilisateur doit être muni d'une webcam avec une bonne résolution + pc</p> <p>Pas de portabilité</p> <p>Algorithmes complexes</p>	

Figure 5 : Tableau descriptif des différentes technologies

A la suite de cette étude, j'ai décidé de réaliser une solution pour ce projet qui se fonde sur la *Reactivation*.

4. Déroulement du projet

Afin de réaliser ce projet, j'ai procédé comme suit :

Au niveau de l'organisation du travail, les tâches sont divisées au cours de l'année pour assurer un bon déroulement du travail.

Une recherche a été effectuée sur les différentes technologies qui peuvent être impliquées afin de répondre au besoin, ce qu'on a appelé la phase d'analyse des besoins ou étude de l'existant. Suite à cette étude, j'ai décidé de travailler avec la solution nommée *Reactivation*.

Il m'a fallu dans une seconde étape commander le matériel nécessaire afin de mettre en place la solution.

5. Planification du projet

Le travail sur ce projet est fourni sur toute l'année dont la première étape est consacrée à préciser l'objectif, la définition des besoins qui repose sur le cahier des charges, l'analyse des besoins afin de décrire la solution proposée. Une fois cette étape achevée, un enchaînement sur la phase de développement doit être réalisée.

Finalement, j'arrive à l'étape de l'intégration du produit et la préparation du rapport ainsi que la soutenance (des livrables intermédiaires seront réalisés tout au long du projet).

J'ai défini aussi à l'aide de *Gantt Project* les différentes tâches de chaque phase, comme illustrée ci-dessous :

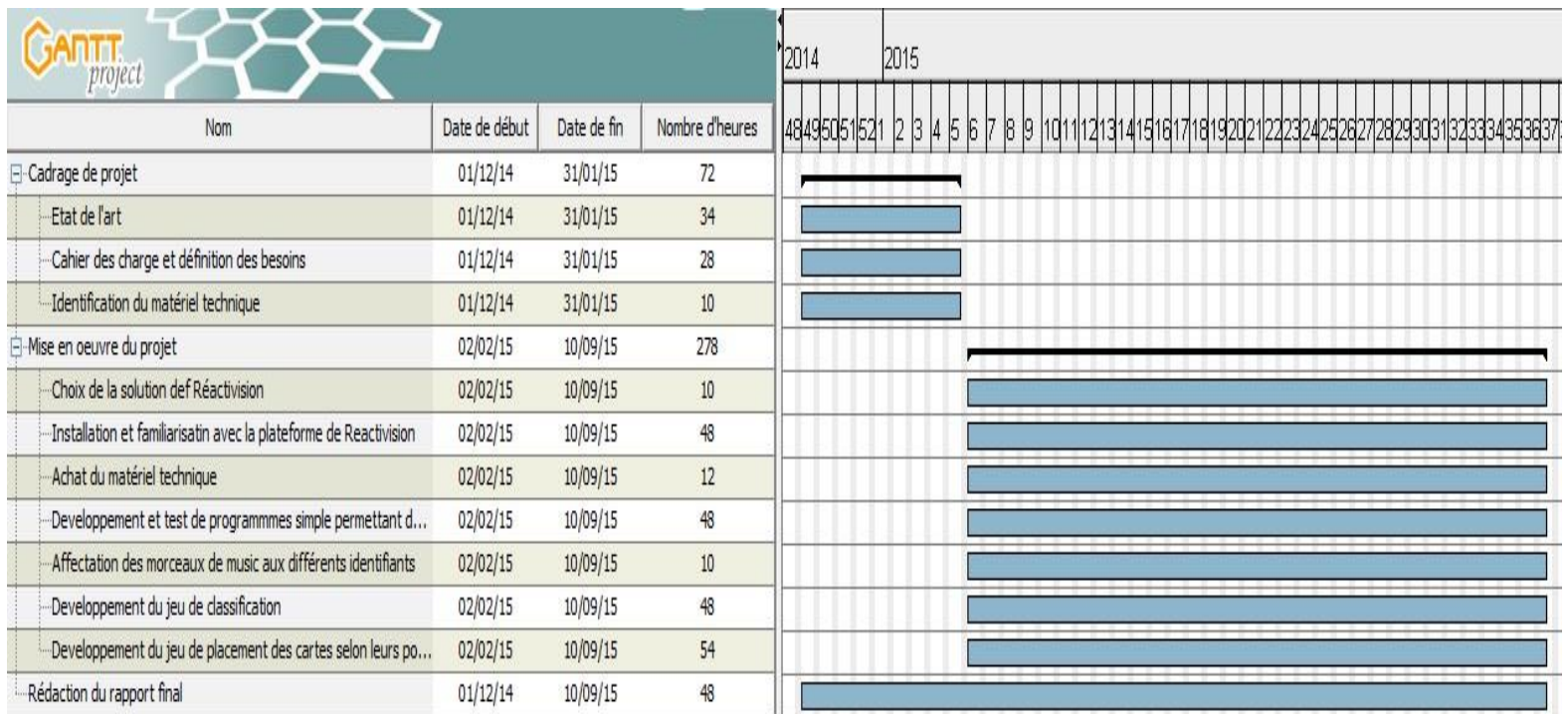


Figure 6 : Planning du projet détaillé Final

III. Description de la solution proposée

Une étude de l'existant a été effectuée afin de limiter le choix à une solution possible pour ce projet. Cette solution est la *Reactivision*.

Ci-dessous, un détail sur la solution choisie et les technologies impliquées.

1. *Reactivision*

1.1 Présentation de la solution

Cette solution qui se base sur la *Reactivision* permet de répondre à la majorité des objectifs du projet. Elle permet aux utilisateurs de jouer avec de simples pièces pour produire l'assemblage du puzzle musical en déplaçant les pièces sur la *Reactable* (elle représente le support physique et l'intermédiaire entre l'application client et l'utilisateur). Au-dessous du support, se trouvera un *Plug Computer* sur lequel tournera mon application.

L'application sera composée du module de la *Reactivision* et l'extension de l'application « client » (qui peut être développée avec différentes plateformes c++, java, c#, PUREDATA ...).

1.2 Architecture

L'architecture de la plateforme de *Reactivision* est donnée comme suit :

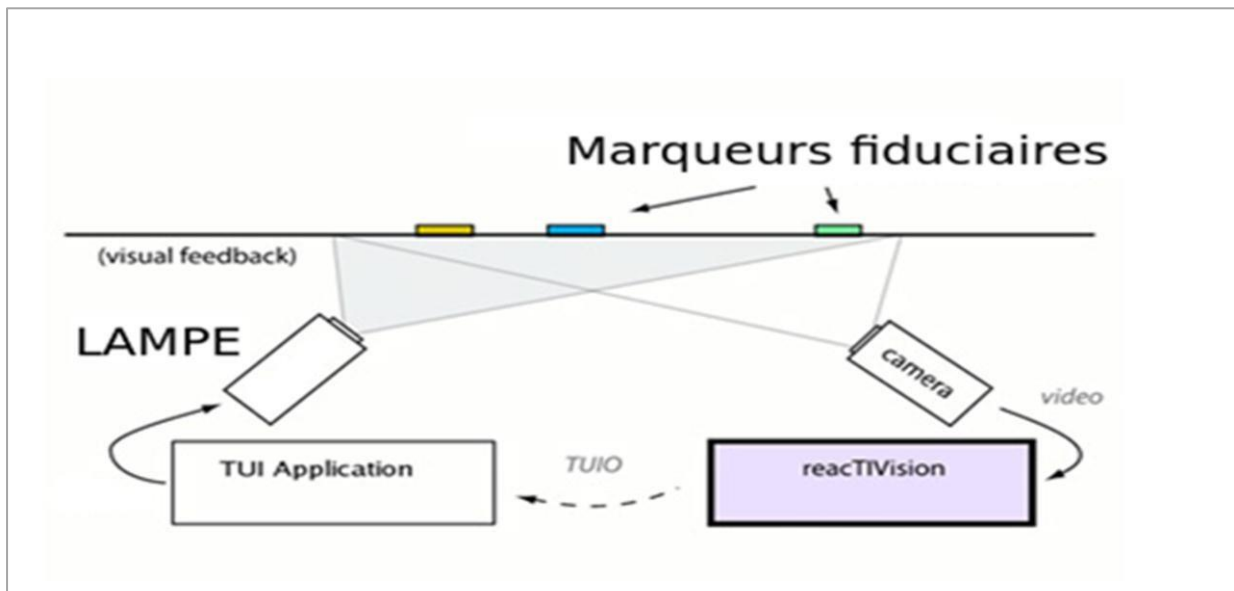


Figure 7 : Schéma de la Reactivision

a) Logiciels

Reactivation :

La *Reactivation*, en elle-même, a été développée comme une composante primaire de la *Réactable*, un instrument de musique électro-acoustique tangible.

Le module *Reactivation* et ses composants ont été mis à disposition sous un ensemble de licences de logiciels *Open Source* (GPL, LGPL, BSD).

Cette application autonome envoie des messages de contrôle *OpenSound* par UDP à toute application client suivie.

TUIO :

TUIO est un protocole permettant la transmission d'une description abstraite de surfaces interactives, y compris les événements tactiles et les états d'objets tangibles.

Ce protocole encode des données de commande à partir d'une demande de poursuite (par exemple, basé sur la vision par ordinateur) et l'envoie à une application client qui est capable de décoder le protocole. Techniquement, TUIO est basée sur *Open Sound Control* - une nouvelle norme pour les environnements interactifs - qui ne sont pas limités au contrôle des instruments de musique.

TUIO Application :

Ce paquet contient deux modules de démonstration et une bibliothèque qui permet au TUIO client (application) de recevoir des messages du module *Reactivation* à l'aide du protocole TUIO. Le module montre les objets et les états de curseur sur l'écran.

Les implémentations de référence TUIO font partie du cadre de *Reactivation* et sont disponibles pour les langages de programmation les plus courants et les environnements médiatiques : C ++, Java, C #, Processing, Pure Data, Max...

PureData:

Pure Data (souvent abrégé Pd) est un logiciel de création multimédia interactive couramment utilisé dans les domaines artistiques, scientifiques et pédagogiques. Sa popularité réside notamment dans sa facilité d'utilisation. Plutôt qu'un langage de programmation textuel, *Pure Data* propose un environnement de programmation graphique dans lequel l'utilisateur est invité à manipuler des icônes représentant des fonctionnalités et à les brancher ensemble.

b) Matériel

Réactable :

La *Ractable* est une interface qui permet de modifier les composantes d'un synthétiseur modulaire. Elle se présente sous la forme d'une surface ronde ou rectangulaire, ayant au centre un point (la sortie du son), sur laquelle on dispose des blocs représentant chacun des éléments du synthétiseur. Les éléments sont reliés entre eux virtuellement comme dans un circuit électrique.

Les joueurs peuvent modifier leurs interactions en faisant varier la distance séparant deux éléments reliés, la fréquence du signal en faisant pivoter l'élément, l'amplitude en déplaçant son doigt autour de l'élément, etc. En s'inspirant de la *Ractable*, j'ai mis en place un support sur lequel l'utilisateur déplacera ses pièces de puzzle et tentera de réaliser les activités proposées. Ce support sera composé d'un plateau cartonné accompagné d'une plaque de PVC (PolyVinyl Chloride).



Figure 8 : Exemple de la *Reactable*

Caméra :

Une caméra située en dessous de la table permet d'analyser la disposition des éléments, ainsi que les mouvements des pièces de puzzle du joueur.



Figure 9 : Type de caméra utilisée

Marqueurs fiduciaux :

La *Reactivision* utilise des marqueurs visuels spécialement conçus qui peuvent être attachés à des objets physiques. Les marqueurs sont reconnus et suivis par un algorithme de vision par ordinateur. Les symboles des marqueurs fiduciaux permettent d'identifier et de distinguer de façon unique les marqueurs ainsi que de soutenir le calcul précis de la position du marqueur et de l'angle de rotation sur un plan 2D.



Figure 10 : Exemple de marqueurs fiduciaux

Lampe :

Un projecteur émet des effets lumineux permettant de pallier un problème du manque de luminosité. Dans un milieu obscur, les activités doivent pouvoir être réalisées en utilisant au mieux les performances de la caméra, donc en utilisant un rayon lumineux efficace.

c) *Prise en main de la solution*

J'ai choisi PUREDATA comme TuioClient, un parmi les applications client, pour développer ma solution, il est principalement un utilitaire pour faire du traitement des signaux sonores et la gestion des instrument musicaux (clavier midi, guitare, etc), il me permet d'avoir plus de détails sur les coordonnées de chaque marqueur fiduciaire tel que son emplacement (sur les axes x ,y et z où z l'angle de rotation) dans les champs de vision du webcam.

A partir de ces coordonnées, je vais pouvoir modaliser la solution pour faire un puzzle musical en divisant les champs de vision sous forme d'une grille.

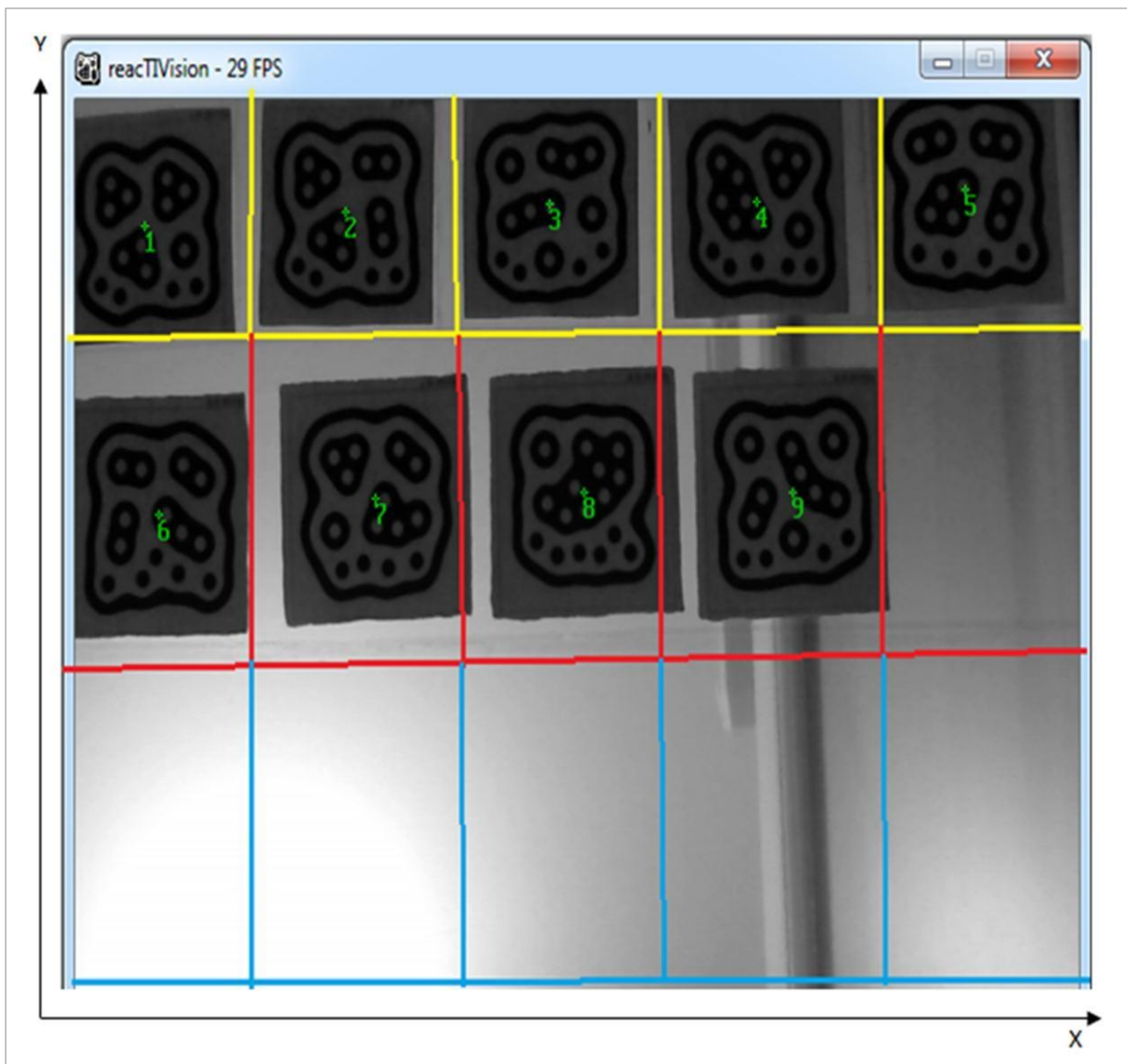


Figure 11 : Champs de vision de *Reactivision* avec deux axes x et y

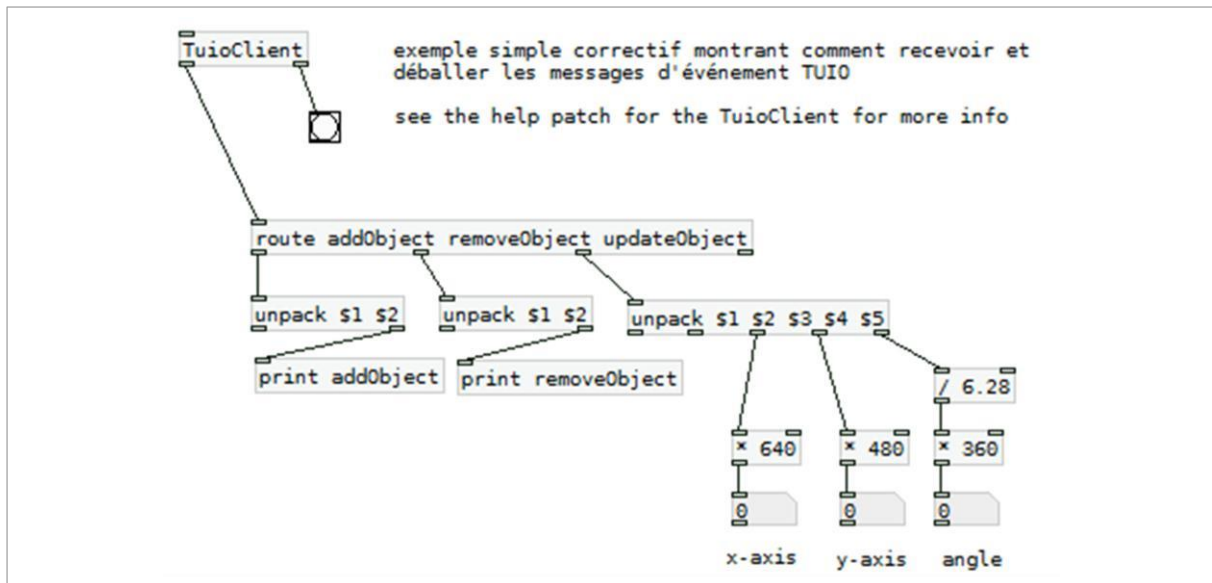



Figure 12 : Recevoir et débiller les informations

Ci-dessous quelques objets utilisés avec la *Puredata*.

Tuioclient : reçoit des messages TUIO au numéro de port fourni, écoute le port 3333 si aucun numéro de port est spécifié :

- Bang  : Le droit de sortie envoie des coups sur les mises à jour du cadre chaque seconde pour indiquer un lien établi.
- « [unpack] » prend une liste et distribue les éléments à ses points de vente.

L'objet [pack] prend une série d'entrées, puis génère une liste concaténée. Par défaut, « [pack] » a deux entrées, chacune d'entre elles accepte un flottant.

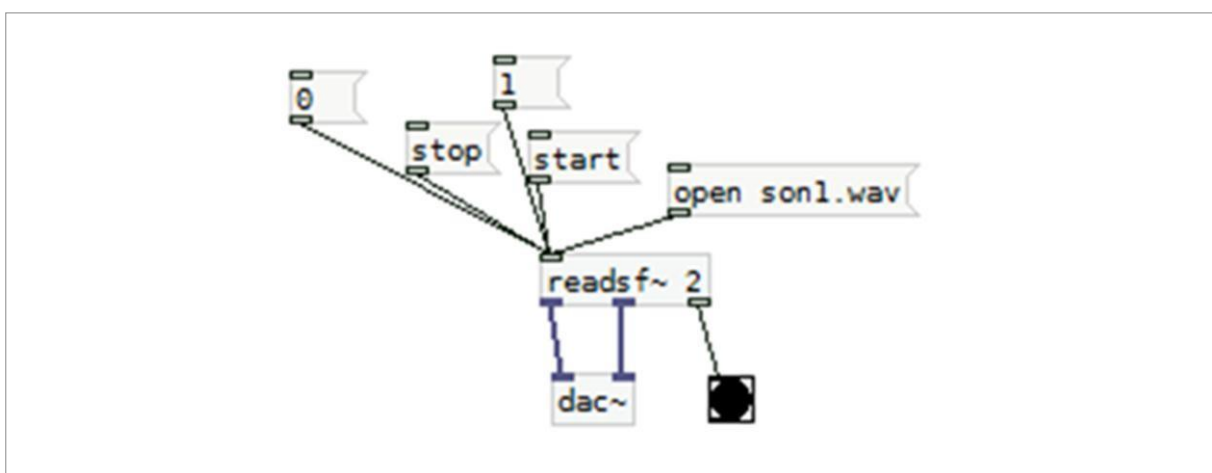


Figure 13 : Exemple de programme "lecture un fichier wav"

L'objet « [readsf ~] » lit un fichier sonore dans ses sorties de signaux. Le fichier sonore doit être ouvert à l'avance (quelques secondes avant que vous en aurez besoin) en utilisant le message «open».

- Message « open » permet d'ouvrir le fichier « son.wav » et le charger en mémoire pour qu'il soit lu après avec la fonction « readsf ».
- Message « start » lance la lecture de musique (on peut le remplacer par le message 1).
- Message « stop » arrête la lecture de musique (on peut le remplacer par le message 0).

L'objet commence immédiatement la lecture du fichier, mais la production n'apparaîtra qu'après l'envoi de « 1 » pour démarrer la lecture. Un « 0 » l'arrête.

La fonction logique est donnée par

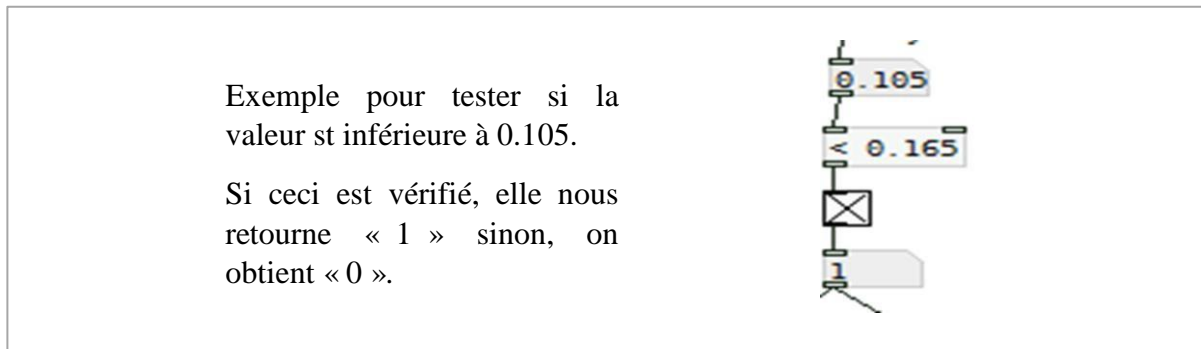


Figure 14 : Exemple pour travailler avec des conditions

1.3 Analyse

Afin de réaliser un jeu en utilisant *PureData* de la section précédente, j'ai joué sur les champs de vision de la webcam comme j'ai montré dans la figure 11 pour la réalisation de deux cas :

a) Cas classification

Dans ce cas, j'ai divisé l'axe (y) par des lignes et chaque ligne est spécifiée à un genre de musique, c'est-à-dire, les marqueurs numérotés de 1 à 3 placés sur la ligne 1 sont affectés à des morceaux de musique (par Exemple Disco), et les marqueurs de 4 à 6 sont affectés à des morceau du genre « Arabe » par exemple. Le cas où on place un marqueur dans une ligne qui ne correspond pas à son emplacement je lance un bip sonore d'erreur.

b) Cas composition

Dans ce cas, on a divisé chaque ligne en 5 colonnes suivant l'axe (x) et on a défini l'emplacement chaque marqueur sur la colonne qui lui correspond, le cas où on le place dans une mauvaise case, on lance un bip sonore d'erreur. La lecture des morceaux ne sera lancée qu'à la seule condition que l'on le mette à sa bonne position et pour garder bien le principe du jeu (puzzle) la lecture des morceaux se fait en mode asynchrone de façon que l'on fait la lecture du morceau 1 puis à la fin de lecture du morceau 1 on passe pour lire le 2 (dans l'ordre jusqu'au morceau 5). A la fin, on annonce la fin de jeu.

Le tableau suivant récapitule les technologies impliquées avec leurs descriptions :

Techniques d'identification	Description	Rayon d'action	PRIX
Reactable	Support pour déposer les différentes pièces Utilisant le dessus d'une table vitrée Basée sur le déplacement d'objets physiques sur la surface de la table	Nécessité d'une webcam avec une bonne résolution Pas de portabilité Complexité des algorithmes et du calibrage des pièces	1 carton d'emballage 62*62*36 cm ~ 5 euros 1 plaque de plexiglass 50*50 cm ~ 10 euros Une webcam avec une qualité correcte HD ~ 30 euro Vidéoprojecteur (Possibilité d'utiliser une lampe de 100w pour éclairer le champ de vision) ~5 euro
Reactivation	Module <i>Open Source</i> multi-plateformes Capture rapide des marqueurs fiduciaux Marqueurs fiduciaux rattachés à des pièces matérielles	Nécessité d'une source d'éclairage pour distinction des pièces (distance, obscurité)	Le module <i>Reactivation</i> → open source L'extension de l'application « client » → open source
Les marqueurs fiduciaux	Sont des objets placés dans le champ de vision d'un système d'imagerie qui apparaît dans l'image produite, pour une utilisation en tant que point de référence ou une mesure		

Figure 15 : Tableau descriptif des Technologies utilisées

IV. Développement des clients *PureData*

1. Jeu de classification

Le jeu de Classification consiste à distinguer à base de l'identification de deux types de musiques différents par exemple « arabe » et « disco ».

La solution est présentée, en premier, sous forme d'arbre de décision que j'ai traduit en programme *Pure-data*.

1.1 Arbre de décision

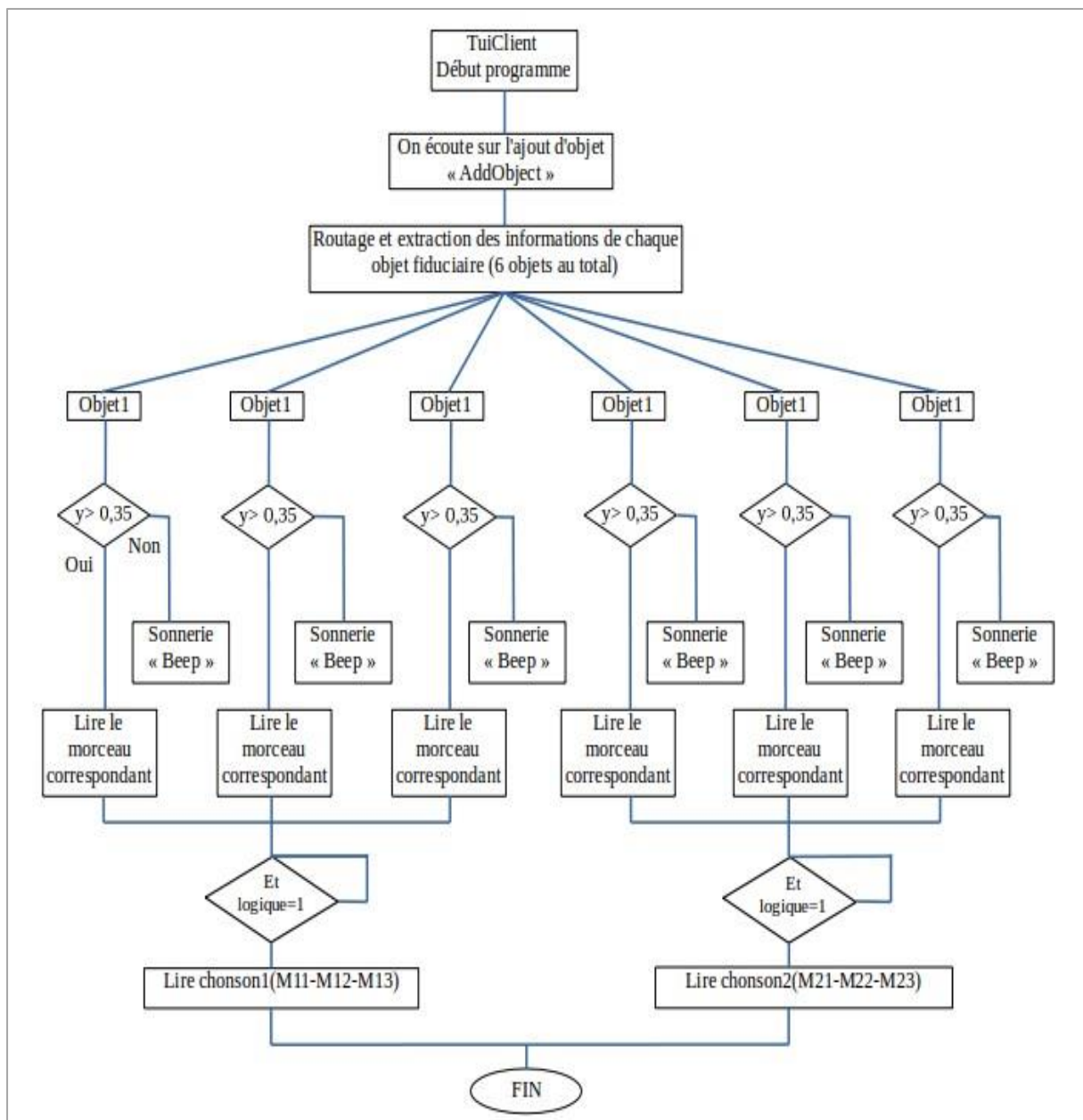


Figure 16 : Arbre de décision du jeu de classification

1.2 Programme *PureData*

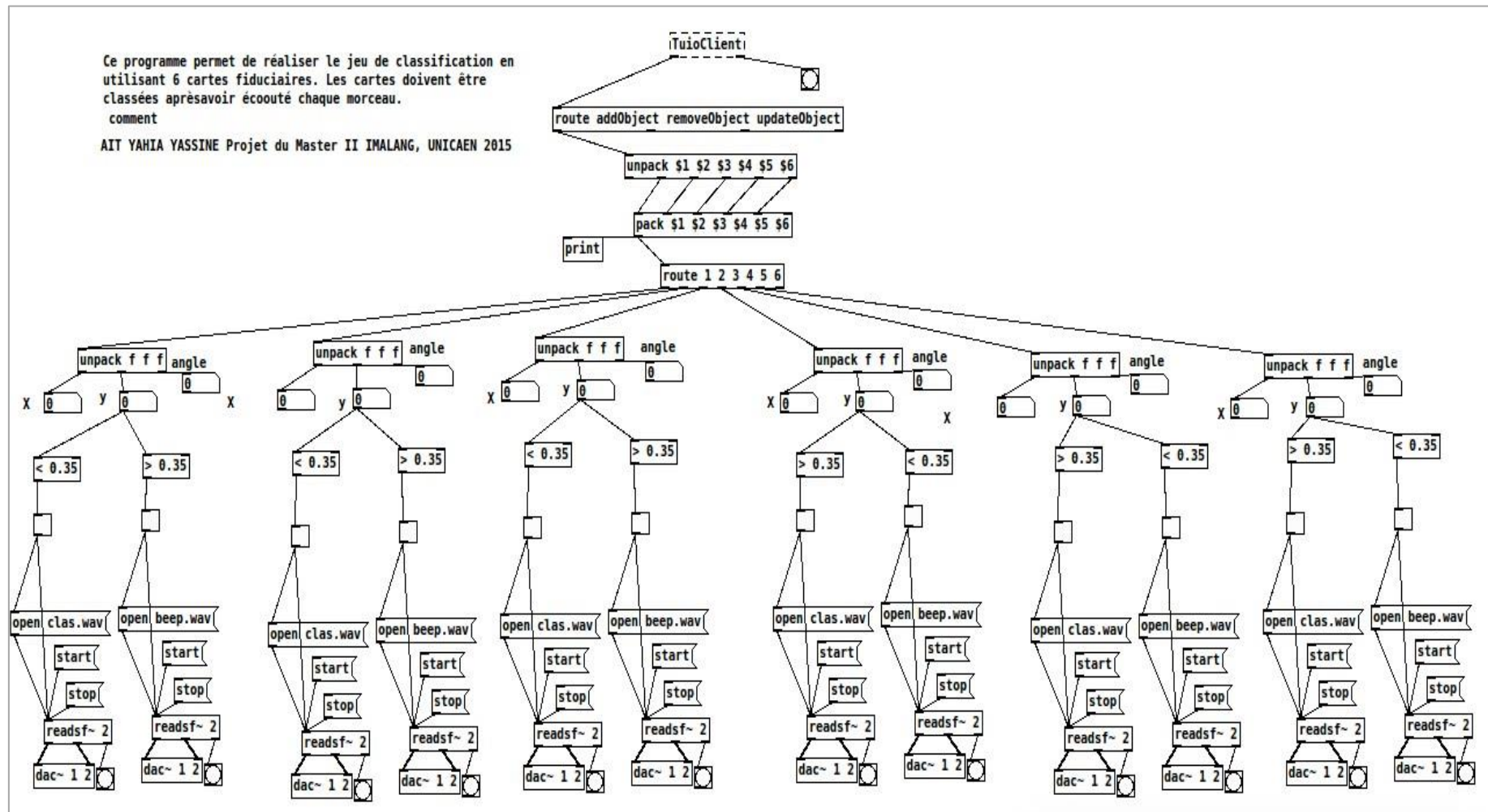


Figure 17 : Programme *PureData* du jeu de classification

2. Jeu de placement des cartes selon leurs positions

L'utilisateur va former son jeu de telle sorte que l'emplacement des pièces va influencer la réussite ou l'échec du jeu.

De la même façon que le programme de classification, le jeu de positionnement est présenté premièrement sous forme d'arbre de décision que j'ai traduit en programme *Pure-data*.

2.1 Arbre de décision

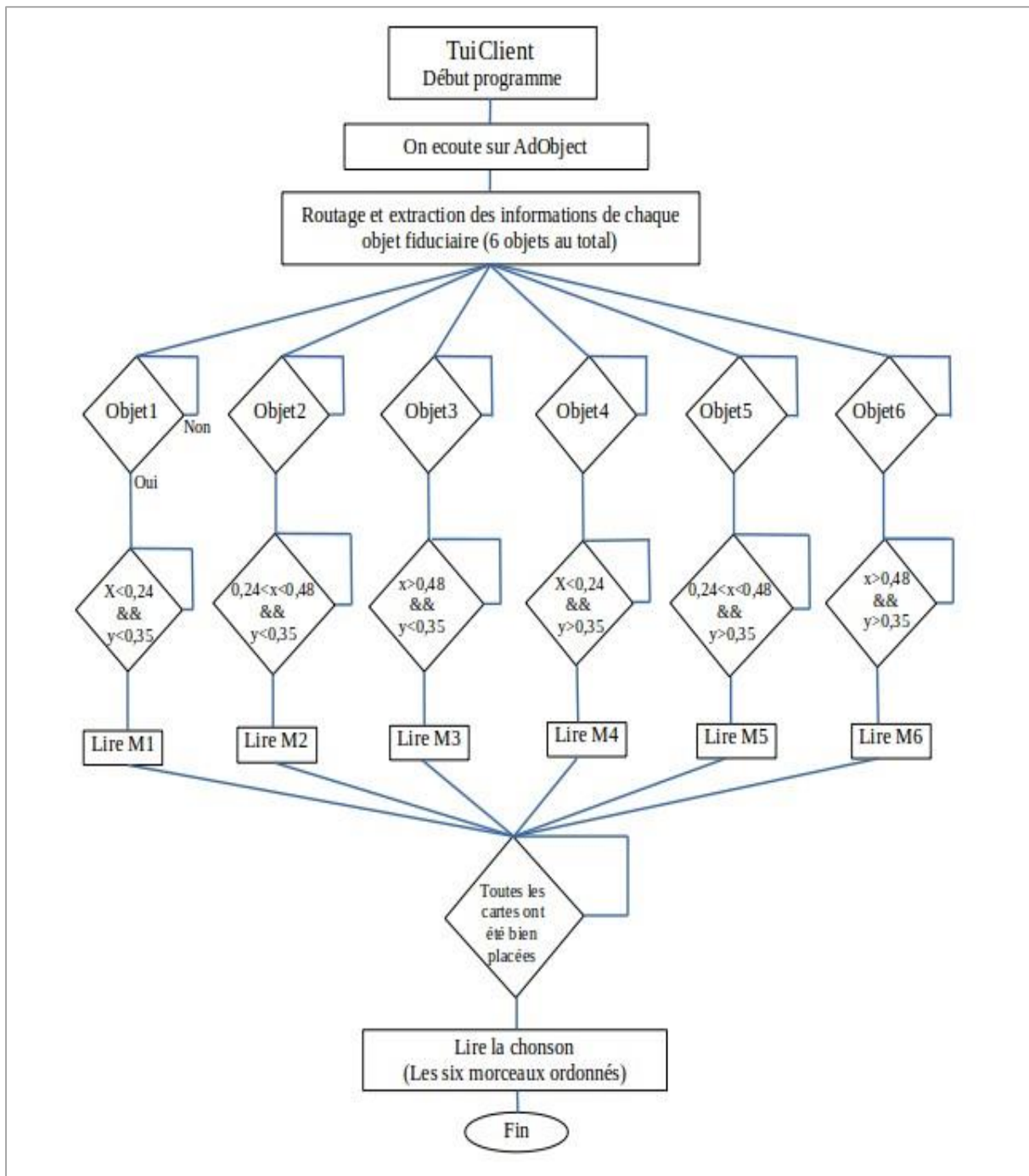


Figure 18 : Arbre de décision du jeu de placement des cartes

2.2 Programme PureData

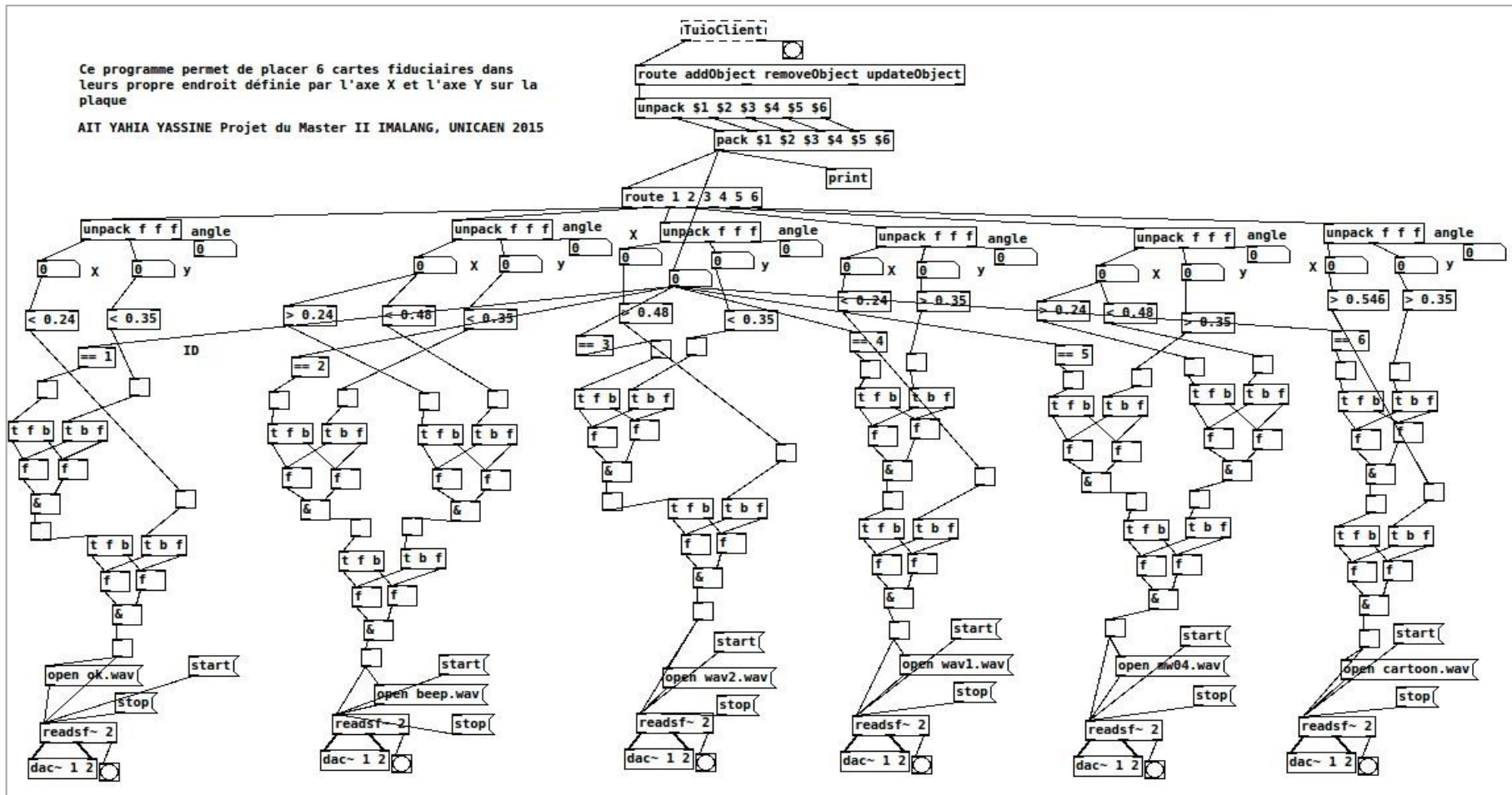


Figure 19 : Programme PureData du jeu de placement des cartes

V. Conclusion

Pour conclure, la solution développée est nommée *Reactivation*, elle se base sur une *Webcam*, une plaque de plexiglass et un ordinateur portable. Cette solution est facile à transporter et son coût total est de 50 euros.

Ce projet a pour but de divertir les utilisateurs et de les rassembler autour d'un même concept. Il existe déjà des jeux qui utilisent la musique afin de faire vivre à l'utilisateur des moments agréables, comme par exemple « l'Auditorium » ou « Audiosurf ». Cependant, ces jeux se contentent d'avoir la possibilité de jouer avec la musique.

Ce produit demandera à l'utilisateur de retrouver un assemblage prédéfini tout au long d'une séance divertissante. Il pourra mettre en avant sa capacité de distinction et de reconnaissance musicales tout en prenant du plaisir.

Finalement, les technologies utilisées et étudiées durant ce projet m'ont permis d'élargir mes connaissances et ma culture.

Abréviations

Ci-dessous, le détail des abréviations utilisées dans ce rapport.

- **API:** Application Programming Interface
- **BSD:** Berkeley Software Distribution
- **GPL:** General Public License
- **LGPL:** Lesser General Public
- **RFID:** Radio Frequency Identification
- **TUIO:** Tangible User Interface Objects
- **UDP:** User Datagram Protocol

Annexe

Sélection d'objets Pure Data Extended 0.43.4

Ouvrir l'aide de l'objet (clic droit + help). La bibliothèque est indiquée entre parenthèses.

Si l'objet ne se crée pas, écrire le nom de la bibliothèque avant le nom de l'objet, suivi d'un « / » comme ceci : [hcs/cursor].

Liste d'objets de base (pd-vanilla) : clic-droit sur page blanche, explorer aussi le Menu Aide > Rechercher.

Autres listes d'objets sur : <http://jeromeabel.net/files/code/pd/docs/>

COMMUNICATION

midout, midin messages Midi (note, controller, program,...)
comport port série
arduino gestion de l'Arduino avec comport (pduino)
lpt port parallèle Linux (zexy)

ENTRÉES

adc~ microphone
pix_video caméra (Gem)
key, keypop, keyname clavier
gemkeyboard clavier (Gem)
MouseState souris
cursor souris (hcs)
gemmouse souris (gem)
hid joystick, teensy (hid)
joystick joystick (hid)
gemtablet tablette (Gem)

SORTIES

dac~ haut-parleurs
output~ contrôleur amélioré
dsp audio on/off (pddp)
gemwin affichage Gem (Gem)

RÉSEAU

netsend, netreceive messages Pd sur le réseau, UDP ou TCP
netclient, netserver *idem* + broadcast (maxlib)
netdist, netrec envoi sur plusieurs [netreceive] (maxlib)
udpreceive, udpsend, udpclient envoi d'octets sur UDP (mrpeach, iemnet)
tcpreceive, tcpsend envoi d'octets sur TCP (mrpeach, iemnet)
tcpclient, tcpserver *idem* (mrpeach, iemnet)
httpreceive, httprec requête HTTP/1.1 (net)
packOSC, unpackOSC construit un message OSC (osc)
sendOSC, dumpOSC, OSCroute *idem* (oscx)
 > voir aussi [nstream~] et [getIP] de xjmmies

RÉSEAU AUDIO

udpreceive~, udpstream~ vecteurs audio sur UDP (net)
streamin~, streamout~ (ext13)
promiscuous~ conversion audio du trafic réseau (ext13)
oggamp~, oggcast~ fichiers ogg (ogg)
mp3streamin~, mp3streamout~ fichiers mp3 (unauthorized)

MÉDIA

--- Midi ---
xeq, midiparse, midiformat fichiers midi (cyclone)
 --- Son ---
soundfiler chargement d'un fichier audio
tabplay~, tabread4~ lecture de tables audio
readsf~, writesf~ fichiers audio sur le disque dur
sfread2~ autre readsf~ (mooslib)
sfwrite~ autre writesf~ (ggee)
readanysf~ lecture de tout format (?)
pdp_mp4player~ fichiers mp4 (pidip) (obsolète)
oggread~, oggwrite~ fichiers ogg (ogg)
wavinfo informations sur le fichier audio (ext13)
soundfile_info *idem* (iemlib)
 --- Image ---
pix_image, pix_multiimage lecture d'images (Gem)
pix_film, pix_movie lecture de vidéos (Gem)
pix_write enregistrement de la fenêtre (Gem)
pix_buffer_read, pix_buffer_write lecture/écriture des pixels en mémoire
pdp_qt, pdp_qt~ lecture de vidéos avec l'audio (pdp)
pix_info informations sur l'image (Gem)
model chargement d'un modèle 3D .obj (Gem)
 > voir (Gem), (pdp), (gridflow)

SYNTHÈSE

--- Son ---
noise~, pink~ générateur de bruits blancs et rose
osc~ oscillateur forme d'onde cosinus
phasor~ oscillateur forme d'onde dent de scie
phasorshot~ *idem* + signal de fin de boucle (tof)
square~ oscillateur forme d'onde carée (hcs)
pwm~ oscillateur PWM (hcs)
tahose4~, ... oscillateur de lecture de table
dirac~ générateur d'échantillons
flite synthèse vocale (moocow)
 --- Image ---
triangle, sphere, rectangle, ... formes géométriques, primitives (Gem)
GEMglBegin chaîne OpenGL (Gem)
pix_set créer un pixel (Gem)
 > voir (Gem), (pmpd), (pdp), (gridflow)

ANALYSE

--- Midi ---
chord reconnaissance d'accord (maxlib)
 > voir (maxlib)
 --- Son ---
fft~ analyse fréquentielle
samplerate~ fréquence d'échantillonnage
env~ intensité sonore
envrms~ *idem* en RMS (zexy)
honk~ détection d'attaque
sigmund~, fiddle~ détection de hauteur, intensité et d'attaque
peakamp~ détection d'intensité (cyclone)
Scope~ visualisation du signal (cyclone)
 --- Image ---
screenize dimension de l'écran (hcs)
pix_data analyse pixel par pixel (Gem)
pix_movement, pix_movement2 analyse des mouvements (Gem)
pix_mean_color couleur d'un pixel (Gem)
pix_blob, pix_multiblob centre de gravité d'une zone (Gem)
pix_blobtracker *idem* + avancé
pix_fiducials détection de formes imprimées (Gem)
pix_background élimination du fond (Gem)
pix_artoolkit utilisation de artoolkit (Gem)
pix_mano analyse des mains (pix_mano)
pix_opencv outils de OpenCV à installer (pix_opencv)
 > voir (Gem), (pdp), (gridflow)

TRAITEMENTS

--- Son ---
vd~, delread~, delwrite~ délais audio
lop~, hip~, hp~, moog~, vcf~, ... filtres audio
biquad~, lowpass~ filtres audio avancés
reson~ résonance (markex)
clip~ contraindre le signal
split~ scinder le signal (sigpack)
limiter~ limiteur (zexy)
dist~ distorsion~ (creb)
freeverb~ réverbération (freeverb~)
plugin~ insertion de plugin LADSPA
vowel~ formants (sigpack)
quantize~ quantification du signal (zexy)
 > voir (pan), (sigpack), (nonauthorized), (timestretch~), ...
 --- Image ---
pix_snap, pix_snap2tex capture d'écran (Gem)
pix_mix, pix_subtract, ... effets sur l'image (Gem)
pix_threshold rasterisation (Gem)
pix_delay, pix_motionblur effets sur le temps (Gem)
rotateXYZ, translateXYZ, ... transformations 3D (Gem)
 > voir les autres objets de (Gem), (gridflow), (pdp)

Jérôme Abel - GNU/GPL - 01.05.2013
<http://jeromeabel.net>

Figure 20 : Les objets PureData

Référence

- http://www.ac-limoges.fr/sti2d/IMG/pdf/RFID_NFC_SIN_FAVARD.pdf
- <http://fr.flossmanuals.net/arduino/>
- <http://fr.flossmanuals.net/arduino/>
- <http://fr.flossmanuals.net/booki/processing/processing.pdf>
- http://www.xrings.net/xrings/article.php3?id_article=383
- <http://www.tuio.org/?software>
- <http://fr.flossmanuals.net/puredata/>
- <http://fr.wikipedia.org/wiki/Reactable>
- <https://www.processing.org/>