en - fr

Un Morpion [Projet ISN Lycée Camille Jullian]

Education
français
Tags: #<Tag:0x007f3b60328f28>

---

engano.isn 2016-01-19 08:58:04 UTC #1

Bonjour à tous,

Nous sommes trois élèves de terminale scientifique spécialité informatique et sciences du numérique en provenance d'un charmant lycée bordelais dénommé Camille Jullian où il fait bon étudier.
Aujourd'hui, nous nous proposons de concevoir un jeu en interaction avec le bras miniaturisé du robot Poppy, Ergo Jr : un morpion interactif. Nous tâcherons donc de rédiger ci-bas un carnet de bord détaillé de notre projet (dont le cahier des charges a été rédigé par l'INRIA).

Ce projet, qui nécessite plusieurs semaines de travail (une dizaine de séances de 2h au moins) alliées à une bonne connaissance des fonctionnalités d'Ergo Jr. Commençons par le début : nous avons suivi une introduction didactique au bras miniature donnée par notre professeur. Cette initiation nous a permis de comprendre ses fonctionnalités de base :

- La programmation par blocs avec snap!
- Les divers actions des moteurs
- Des capteurs
- Les branchements et l'initialisation

Nous avons ensuite suivi une activité de groupe proposée par l'INRIA.

[Rédaction de la suite à la séance suivante - à suivre...]

Activité : Ergo Jr joue à Tic-Tac-Toe

Activité Ergo Jr : Tic-Tac-Toe (work in progress)

Lycée Camille Julian

Photos et vidéos des projets réalisés avec la plateforme robotique Poppy

---

Gallery of Snap! projects used with Poppy robots

---

engano.isn 2016-01-26 08:52:59 UTC #2

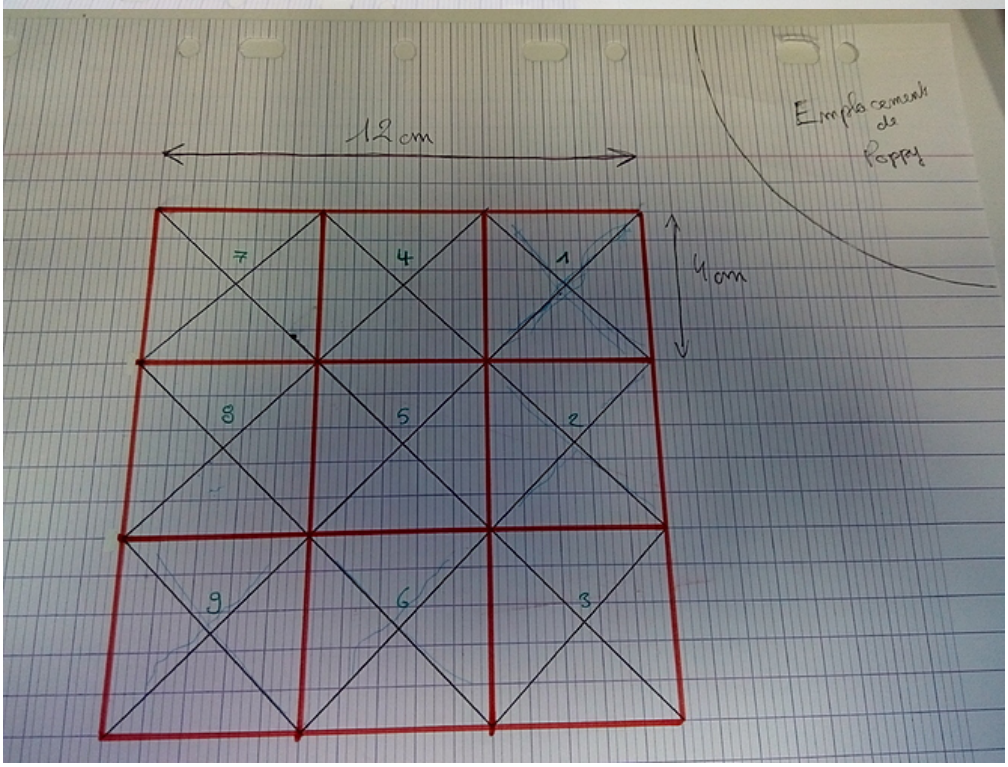
Séance 2 :

Nous avons continué l'initiation à l'application de programmation par blocs Snap! (unit 1). Au cours de celle-ci, nous avons appris à relever les coordonnées angulaires de chaque moteur, et nous sommes familiarisé avec les différentes fonctions du bras robotisé. Nous avons notamment programmé un mouvement où le bras tente des approches sociales affiliées à l'être humain : un signe "bonjour" et une danse conceptuelle (aléatoire) ont été programmés.
Nous avons particulièrement apprécié cette activité, à la fois intéressante et simple d'accès, notamment grâce à l'interface simplifiée de programmation Snap! et le système de coordonnées des moteurs intuitives (grâce à la fonction get_position en mode Compliant, il est possible de récupérer les données angulaires).

engano.isn 2016-02-09 08:57:26 UTC #3

Séance 3:

Nous avons commencé le projet Tic-Tac-Toe Ergo Jr. , nous avons fait la séance 1 ("Setup d'expérimentation"), consistant à construire la grille de jeu de façon la plus ergonomique possible pour le bras robot, afin qu'il puisse effectuer tout les déplacements pour indiquer dans quelle case il veut jouer. Nous avons donc choisi de positionner le bras dans un angle de la feuille, à la diagonale de la grille. La grille a elle une taille de 12cm sur 12cm et des cases carrées de 4cm de côté. Ci joint deux photos illustrant la grille et le bras robot.



La grille construite, nous nous sommes penchés sur la détection des cases par la webcam intégrée au bras robotisé. Pour ce faire, nous avons recours à la librairie OpenCV disponible sur Processing, dont l'algorithme de détection de contours va nous permettre de calculer les dimensions approximatives de la grille. L'algorithme étant déjà rédigé, il n'y a plus qu'à ouvrir une image avec le programme, et tels en sont les contours selon la détection algorithmique :

(Les algorithmes utilisés sont disponibles à cette adresse : **https://github.com/atduskgreg/opencv-processing**)

Stephanie 2016-02-12 17:13:36 UTC #4

Merci pour ce partage. C'est super, ça avance vite !
Avez-vous connu des difficultés ?
Avez-vous déjà essayé de faire dessiner des croix et des ronds à l'Ergo ? Si oui, cela fonctionne-t-il bien ?

---

engano.isn 2016-03-01 07:43:49 UTC #5

Bonjour nous n'avons pas connu de réelles difficultés pour le moment mais dorénavant nous entrons dans le cœur du projet et dans la partie programmation qui risque nous donner plus de difficultés et de fil à retordre.
Et nous ne nous sommes pas encore pencher sur la partie dessin des des croix et des ronds par l'Ergo nous entamons tout juste la partie déplacements du bras sur le plateau de jeu et le pointage du bras sur la case souhaitée dans le code.

---

engano.isn 2016-03-01 09:00:50 UTC #6

Bonjour, aujourd'hui nous avons réparti les taches en trois parties.
Tout d'abord l'un d'entre nous s'est occupé de l'algorithme de détection des contours de la grille, avec une difficulté supplémentaire : l'utilisation d'une webcam était requis. Il a donc fallu allier l'algo de détection des contours de la grille à celui-ci contrôlant l'affichage des images capturées par la webcam. Il faut donc pouvoir faire une capture d'écran pertinente (contenant la grille et le contenu des cases) avec une fonction saveFrame(), pour ensuite détecter le centre de chacune des cases. Ce n'est pas une mince affaire...
Ensuite le second à commencer à coder sur snap pour déplacer le bras sur chaque case du plateau de jeu en utilisant les coordonnées des moteurs et créer des blocs pour chaque actions avec un retour à une position de base. On s'est rendu compte qu'il y avait un problème avec la stabilité du plateau de jeu quand celui ci est une feuille de papier il faut donc réfléchir à résoudre ce problème (un plateau plus solide d'un matériau différent peut être).
Ci joint une vidéo illustrant le mouvement du bras robot sur la case 1 après appui sur la touche 1 du clavier:

Enfin le dernier a commencé à réfléchir aux règles du jeu imposées et à la manière de les transformer en algorithme afin de les programmer à partir d'un ordinateur, tout en tenant compte des contraintes numériques et physiques de l'Ergo Junior.

---

engano.isn 2016-03-13 19:42:09 UTC #7

Bonjour,

Veuillez d'ores et déjà avoir l'amabilité de nous excuser pour la réponse plus que tardive et notre manque relatif d'activité sur ce sujet...

Mardi 08 mars : Nous avons continué les tâches sur lesquelles nous nous étions penchés la semaine précédente.

Tout comme il en avait été question le 1er mars, l'un des membres du groupe a continué de travailler sur l'algorithme de détection vidéo. La tâche à effectuer restait dans la continuité de ce sur quoi il avait réfléchi auparavant : la transcription de l'algorithme OpenCV dans l'algorithme de gestion de webcam.
En gros, cela a consisté à recopier, ligne par ligne, le code OpenCV dans l'autre, en testant à chaque fois que le programme compilait correctement. Première difficulté : faire "comprendre" à l'algorithme qu'il doit récupérer les informations directement d'un flux vidéo live (objet "cam") et non pas d'une simple image source (objet "src"). il faut donc réécrire certaines commandes, mais là n'est encore n'est pas la tâche la plus compliquée... Ce qui donne vraiment du fil à retordre est la copie d'écran : en effet, comme on l'a expliqué la semaine passée, il faut que l'algorithme de détection de contours de polygones puisse travailler sur des images, car on ne peut pas utiliser

directement les informations d'un flux vidéo (celles-ci ne sont pas stockées dans une variable). On utilise donc la fonction copy() pour sauvegarder les pixels qui nous intéressent dans une variable PImage (et non pas saveFrame(), qui stocke dans le dossier du programme chacune des prises d'écran, ce qui est très lourd pour la mémoire). Là encore, on travaille sur l'image de la grille imprimée, réalisée avec GIMP en noir sur blanc afin que la détection soit optimale.

Un autre élève à lui continuer la partie déplacement du bras robot sur la grille pour indiquer sur quelle case jouer. Cette partie ne pose pas de gros problème mais est plutôt longue à rédiger sur snap. Le seul petit problème est la stabilité du plateau de jeu qui sera réglé en fixant les éléments de jeu grille et bras robot sur un plateau rigide. Cette partie sera terminé normalement à la prochaine séance et l'élève en question sur penchera alors sur les règles du jeu et comment les implantées informatiquement, car cette partie nécessite une plus grande réflexion en vue de sa difficulté.

Le dernier élève s'est quant à lui rendu compte des contraintes techniques et numériques qui risquent de poser problème dans la continuité du projet. Tout d'abord le bras du robot n'est pas extensible ce qui nous oblige à travailler sur un diamètre bien précis. Numériquement, les contraintes sont que l'on utilise plusieurs technologies (robotique, vision) et qu'il va falloir faire le lien entre elles, c'est à dire transmettre les données Processing à Snap!. Ensuite ce même élève a commencé à mettre en place un algorithme permettant d'avoir une vision globale de la future programmation des règles du jeu sur Processing.

---

engano.isn 2016-03-20 13:21:09 UTC #8

Bonjour,

Mardi 15 mars : Nous avons continué finalisé certaines tâches sur lesquelles nous nous étions penchés la semaine précédente et avancé dans d'autres.

En continuant sur le déplacement du bras robot l'un des membres du groupe à finaliser les mouvements du bras sur le plateau de jeu. Le robot est maintenant capable de se déplacer sur la case souhaitée. Pour l'instant l'ordre de déplacement est désigné par le clavier par exemple la touche 1 fait déplacer le bras sur la case 1. Cet élève s'est ensuite penché sur la question des règles du jeu et comment les implanter dans un code sur processing. Cette partie va nécessiter une plus grande réflexion et un approfondissement des connaissances sur processing et sur l'utilisation d'un tableau. Pour cette partie des recherches personnelles seront donc nécessaires afin d'aboutir à la création du jeu et de son interface. Cette recherche personnelle se fera en autonomie en dehors des cours d'ISN. Nous avons aussi commencé à faire l'inventaire des fonctions Snap et processing utilisées pour nous guider dans la réalisation du projet. Et pour illustrer les déplacements du bras robot et du code Snap utilisé voici une video (déplacements du bras) et une photo (code Snap):

Un autre élève a quant à lui continué de travailler sur l'algorithme de détection des contours avec la webcam, petit à petit, difficilement. C'est une tâche fastidieuse où, encore une fois, il faut retranscrire dans l'algorithme de gestion du flux vidéo les commandes afférant à la détection vidéo, ligne par ligne...
En parallèle, le carnet de bord, dressant l'historique des étapes et des difficultés rencontrées au cours du projet a commencé à être rédigé. Il indique notamment la provenance des ressources utilisées (d'où vient notre matériel, d'où sortent les algorithmes utilisés...) ainsi que la signification de chacune des fonctions utilisées dans les algorithmes (leur utilité, ce qu'elle renvoie...).

Le dernier élevé a déterminé les fonctions à programmer sur processing nécessaires au bon fonctionnement du jeu. Celui-ci s'est penché sur une fonction InitialisationJeu permettant d'initialiser la partie afin d'en recommencer une nouvelle. Malgré quelques difficultés rencontrées pour mettre les commandes dans une fonction, le but à été atteint.

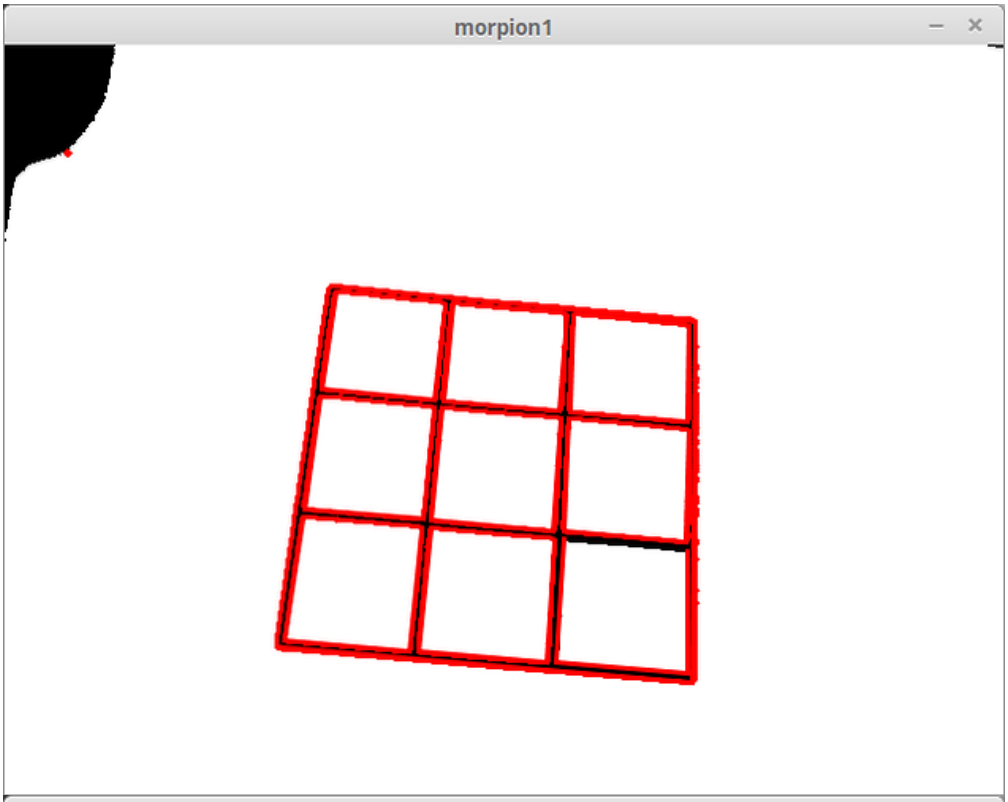
---

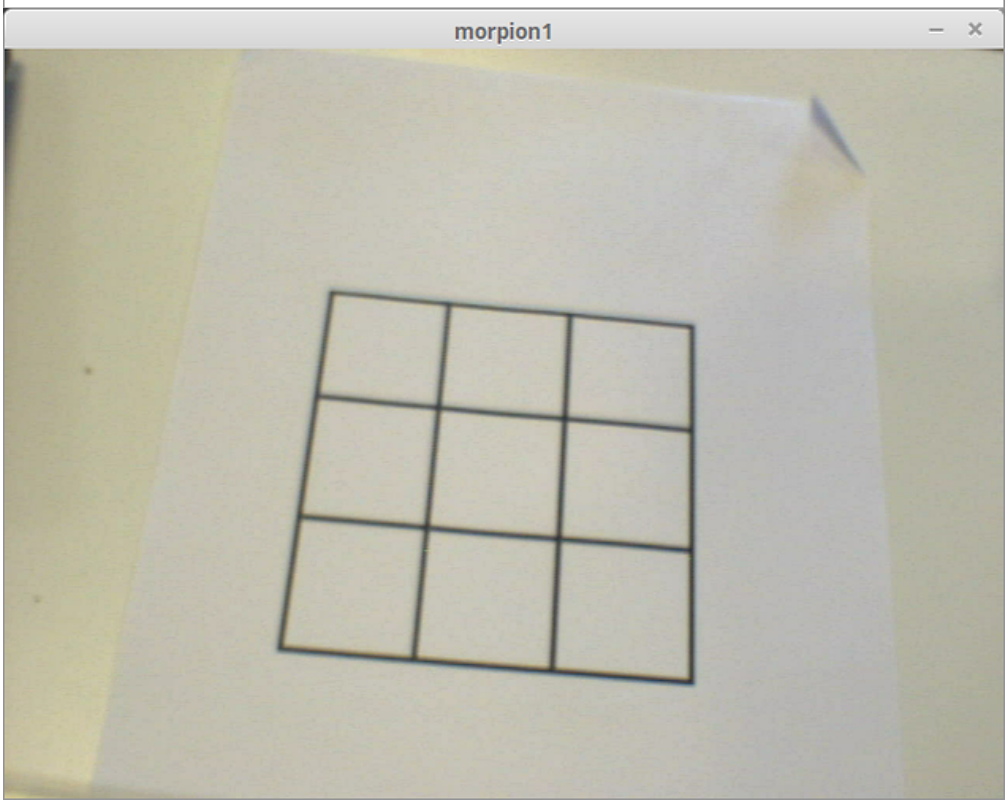
engano.isn 2016-03-22 08:55:47 UTC #10

L'algorithme de détections des contours fonctionne bel et bien !
Après une intervention du professeur qui nous est venu en aide, les quelques points sur lesquels nous bloquions ont été levés : le programme a été revu, corrigé et simplifié.
Tout d'abord, l'algo a été séparé en trois onglets :

- Un onglet **MORPION1**, contenant le *setup()*, l'importation des librairies, la fonction *draw()* qui détecte si une webcam a été branchée, la déclaration des listes de contours...
- Un onglet **DETECTION**, contenant la détection des contours, l'affichage de la surface maximale des aires détectées, le nombre de contours détectés... Cet onglet contient aussi la fonction binarisation, qui va traduire les couleurs de l'image récupérée par le flux vidéo en niveau de gris : un avantage notable pour une meilleure détection des contours.

- Un onglet **TEST** qui contient une fonction *keyPressed()* faisant appel à la fonction binarisation lorsque la touche 's' est pressée, ainsi que l'affichage des contours sous la forme de polygones.





L'élève s'étant occupé de la partie déplacement du bras robot à commencer à rédiger une partie du code visant à implémenter les règles du jeu tout en reliant les différentes parties externes tel que le bras la caméra détection dans le code. Pour savoir quand faire intervenir ces parties. L'élève qui se penché à la base sur cette partie était absent au dernier cours les avancements sur cette partie n'ont pas été très poussé car nous n'avions pas pu récupérer son travail fait précédemment. Notre objectif est donc de réussir à coder le jeu dans un délai plutôt cours afin de bien se répartir les tâches du codage sur processing pour produire un premier rendu le plus vite possible pour se rendre compte des différents bugs et perfectibilités ou oublis..

engano.isn 2016-04-05 07:57:07 UTC #11

Résumé des séances du 29/03 et 25/03:

Deux élèves ont collaboré pour la partie code sur processing. Ils ont programmé une fonction test du jeu afin de pouvoir créer un code permettant à l'ordinateur de jouer aléatoirement, de façon a repérer les cases vides pour jouer sur case non occupé. Les choix du joueur sont définis volontairement et seront codés ultérieurement en lien avec Snap! et le bras robot afin que le joueur puisses choisir sur quelle case jouer en utilisant par exemple une touche du clavier. Ensuite ils se sont penchés sur le code d'un début d'intelligence artificielle. Cette partie du code est actuellement en réflexion il faut que nous arrivions à implémenter une fonction qui permette à l'ordinateur de déduire si le joueur à la possibilité de gagner en jouant sur une certaine case, afin que l'ordinateur puisses contrer le joueur en jouant lui sur la case en question. Il faut aussi coder une fonction qui permettrait de détecter sur quelle case l'ordinateur pourrait jouer pour gagner. Et si l'ordinateur ne peut ni contrer ni gagner il jouerait aléatoirement. Au cours la dernière séance les deux élèves ont remarqué que la difficulté portera plus sur la rédaction de la fonction portant sur le coup gagnant de l'ordinateur sachant que pour contrer le joueur ils pourront se servir du code de la détection de la partie gagnée. C'est donc sur ceci que portera le travail de ces deux élèves et l'objectif étant que cette partie soit finie au retour des vacances.

(à compléter) (suite en rédaction)

---

engano.isn 2016-04-26 07:57:31 UTC #12

Suite à un coup du destin en notre faveur, le projet fit, au cours des vacances, un pas de géant.

- La méthode régissant la détection des contours a été finalisée, le centre de chacune des cases a été trié de sorte à apparaître dans une ArrayList() par ordre décroissant en ordonnée et croissant en abscisse afin d'obtenir la grille de morpion suivante :
1	2	3
4	5	6
7	8	9

- Dans le même programme ont été réunis l'algorithme de détection des contours et celui des règles du jeu.

- Une interface HTTP a été créée pour faire le lien entre la plateforme web Snap! et Processing, notamment afin de transmettre à Snap! la case sur laquelle le robot souhaite placer un pion.

Il faut à présent parvenir à communiquer à Snap! la position de la meilleure case sur laquelle jouer, déduite à partir d'un algorithme...

---

engano.isn 2016-05-10 07:26:03 UTC #13

Bonjour,
Nous sommes désolés de ne pas avoir posté nos avancements depuis un certain moment.
Au retour des vacances nous avions fais des recherches concernant l'intelligence artificielle. Mais malheureusement nous avons eu beaucoup de mal à coder cette intelligence. Nous ne sommes donc pas parvenu à développer cette partie là même si nous avions proposer des choses qui au final se sont vu pas assez poussé. La difficulté fut donc trop grande. Le professeur à donc décider suite à notre blocage sur les différentes parties du projet de nous fournir la partie code de processing. Nous nous sommes rendus compte que nous n'avions pas les compétences et le niveau pour parvenir à finaliser le code processing.
Nous avons travailler aussi sur la partie Snap! qui deplacerait le robot sur la case souahitée par le jeu mais nous nous sommes aperçu de la présence d'une défaillance dans le code processing quand l'ordinateur doit définir le meilleur coup. Comme la resolution de ce problème nécessite la caméra nous n'avons pas pu nous pencher au final sur la partie snap!, déplacement du bras en focntion du code jeu.

(fin de rédaction en cours)

---

engano.isn 2016-05-25 16:46:19 UTC #14

Bonjour,
Je poste une vidéo montrant une partie de morpion contre le bras robot Ergo Jr.
Nous avons donc réussi avec l'aide du professeur à pouvoir finaliser le projet.
Ce qui nous permet de jouer une partie de morpion contre le bras robot.
Il nous resterai juste à peaufiner l'interface et régler quelques bugs de jeu pour réellement finaliser le projet.
Lien de la vidéo :

---

christophe 2016-05-25 19:38:11 UTC #15

Voilà une vidéo pour Didier et les robotscars

Stephanie 2016-05-26 14:58:47 UTC #16

En effet, la vidéo est très bien faite et tout est très bien expliqué.
Voici le lien pour participer : **http://dm1r.fr/festival-roboscars/**

Félicitations à vous ! Je suis certaine que vous allez impressionner le jury 🙂

---

engano.isn 2016-05-31 07:06:42 UTC #17

L'inscription pour le concours a été faite.
