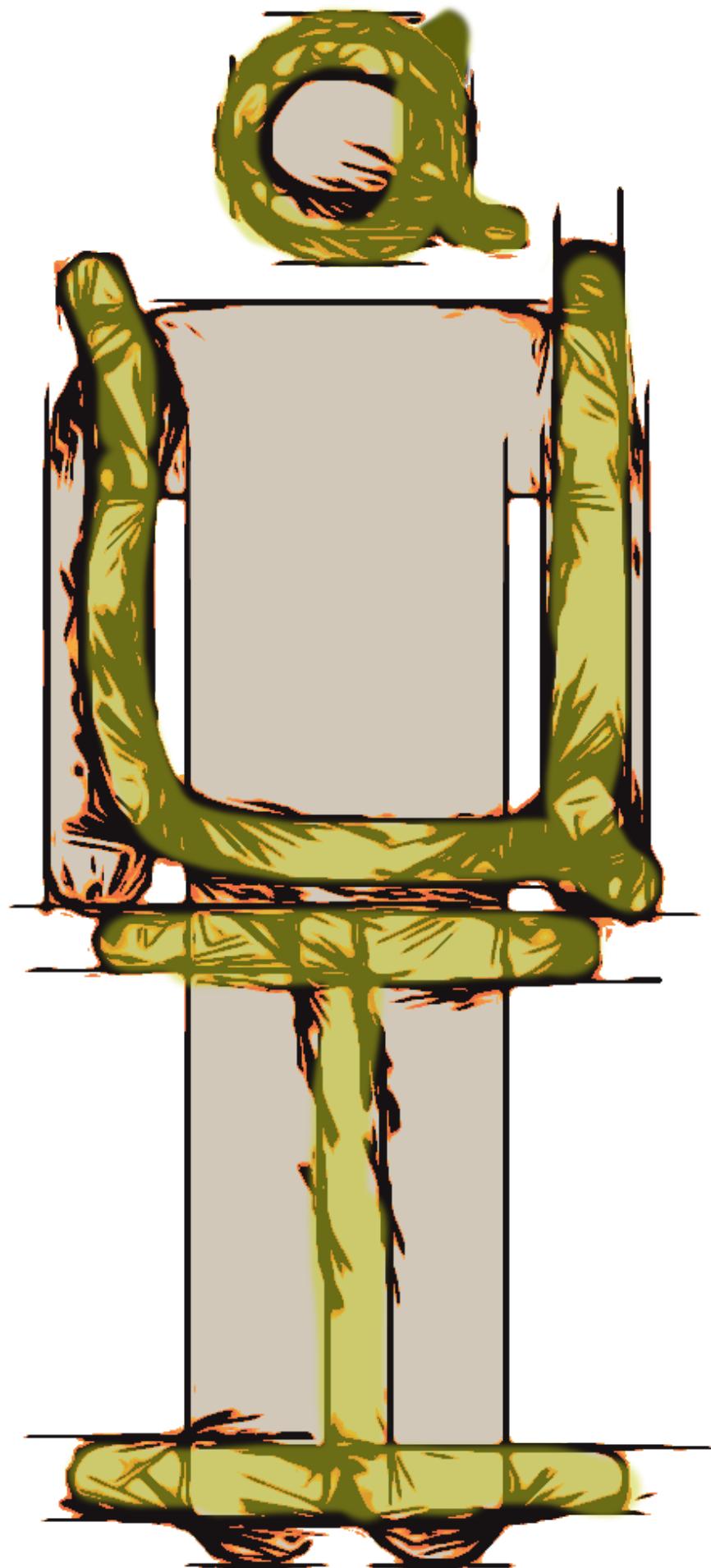


AUI, Une Interface Utilisateur (A User Interface) (pour les canevas infinis)



Résumé rapide :

- La volonté de créer des exercices interactifs pour les élèves autour de la manipulation des nombres et des opérations amène à une interface utilisateur particulière.
- La manipulation passe principalement par la mise en contact ("Touché !") d'un objet et de l'opérateur qui va le transformer.
- Tout ce qui a été conçu pour manipuler les nombres est généralisable à n'importe quel type d'objets.

Introduction :

Utiliser un canevas infini donne, à mesure que l'on se l'approprie, des idées de création qui demandent une interface utilisateur adaptée. L'interface présentée ici découle des principes suivants:

- pas de menus: à l'écran, il y a le contenu et le curseur, et c'est tout
- le clavier et la souris sont des instruments très satisfaisants pour les mains humaines, car ils sont de bons conducteurs de subtilité
- l'interface est une collection imbriquée et grandissante de concepts autour du canevas infini

Les raccourcis clavier/souris pourront changer, les couleurs, le style pourront changer mais l'important ici est la formalisation.

Cette formalisation permet:

- de créer des liens entre les différents concepts mis au jour
- au développeur de commencer à réaliser ce qu'il a imaginé (bien qu'il en découvre encore *en faisant les choses*)
- de construire un langage commun pour les utilisateurs/développeurs par exemple, les concepts, une fois appropriés, peuvent être invoqués en classe sur *papier*

Cette interface va à contre-courant de (ou au moins vient en décalage par rapport à l') évolution de l'utilisation des ordinateurs. Depuis les écrans tactiles, on fait comme si l'utilisateur n'avait que 2 doigts (ou au mieux 4). Avec Une Interface Utilisateur (AUI), les mains des utilisateurs, chacune ayant plus d'un doigt (et c'est bien heureux), leurs permettent, au fur et à mesure, d'être comme des joueurs de violon, de plus en plus proche de leur instrument, jouant avec leur propre style, comme chacun a sa propre graphie avec un stylo sur le papier. Pas besoin de matériel coûteux: on revient au clavier/souris basiques, avec des ordinateurs normaux (pas besoin de surpuissance), et un navigateur internet standard (qui fait office de machine virtuelle avec un affichage), sans obligatoirement avoir besoin d'être connecté à Internet.

Le caractère fondamental des concepts découle du caractère fondamental de la manipulation/transformation d'objets (c'est de la "chimie virtuelle"). Par conséquent, les concepts présentées peuvent sembler être des découvertes "faciles". Au fur et à mesure, les nouvelles découvertes seront plus sophistiquées et sûrement moins fondamentales. Les raccourcis clavier/souris (multi-digitaux) pourront être transposés à des appareillages techniques plus modernes quand ceux-ci verront le jour (contrôle par la pensée).

La portée éducative du logiciel est sa raison d'être.

- Premièrement, elle permet au professeur de présenter de manière interactive et animée des concepts, sans avoir recours aux vidéos, bien trop monolithiques, en suivant son propre style. Elle permet à l'élève de découvrir des concepts, en autonomie, par des expériences "sans conséquence" (sur sa moyenne sacrée) : On essaie, on verra bien ce que ça donne.
- Deuxièmement, elle pourrait enfin permettre à des groupes d'élèves de travailler en coopération sur des projets divers (pas forcément mathématiques), en continuant à utiliser l'interface qu'ils se seront appropriée, voire de créer leurs petits mondes (comme quand les enfants jouent aux Lego et aux PlayMobil, ou à Minecraft).

La formalisation qui accompagne cette interface permet aux professeurs d'exprimer facilement leurs idées, puis de les transmettre dans un format générique, pour éventuellement laisser les devs les réaliser ultérieurement.

Seuls les concepts les plus forts vont être présentés (même si les autres restent accessibles au lecteur curieux).

-Clavier/souris ? Sérieux ?!

- C'est une étape de transition avant que l'on puisse manipuler sans les mains ("psynipuler").
- Pour l'instant, de la même façon qu'une bonne partie de l'humanité utilise, depuis quelques siècles, le couteau et la fourchette pour manger, ce sont des instruments suffisamment conviviaux pour qu'ils soient utilisés comme des prises en escalade, pour se hisser techniquement. Avec les interfaces tactiles, les doigts, sur l'écran, cachent l'écran, et comme un prof qui resterait tout le temps devant le tableau, c'est vite pénible.

L'interface utilisateur étant principalement visuelle, dans ce livre, des illustrations vont naturellement accompagner les concepts.

Les concepts présentés semblent aller de soi ("Pourquoi parler de choses si évidentes ?"), mais ils constituent une nouvelle façon de véhiculer les intentions des utilisateurs (qu'ils soient devant l'écran ("simples" utilisateurs) ou "derrière" l'écran (concepteurs/développeurs)). L'interface se veut un moyen pour les utilisateurs de manipuler, de créer, d'essayer des choses, c'est à dire de renouer un lien sain avec la machine, qui soit en continuité avec les manipulations que fait spontanément l'enfant en train de se développer.

Une implémentation des concepts présentés dans cet ouvrage a été commencée. La confrontation des idées avec leur réalisation logicielle est très féconde. C'est souvent sur l'établi qu'on a des nouvelles idées, dont la formalisation donne de nouvelles idées, dont la réalisation donnera de nouvelles idées, etc. Pour être clair : le logiciel entamé n'a pas toutes les fonctionnalités du livre, et, il reste encore beaucoup de concepts à découvrir. Le logiciel qui a permis l'excavation des concepts est davantage un laboratoire qu'un supermarché. C'est un grand monolithe d'exploration (gros bloc de code informatique aux intentions variées) qui a été conçu pour une utilisation hors ligne.

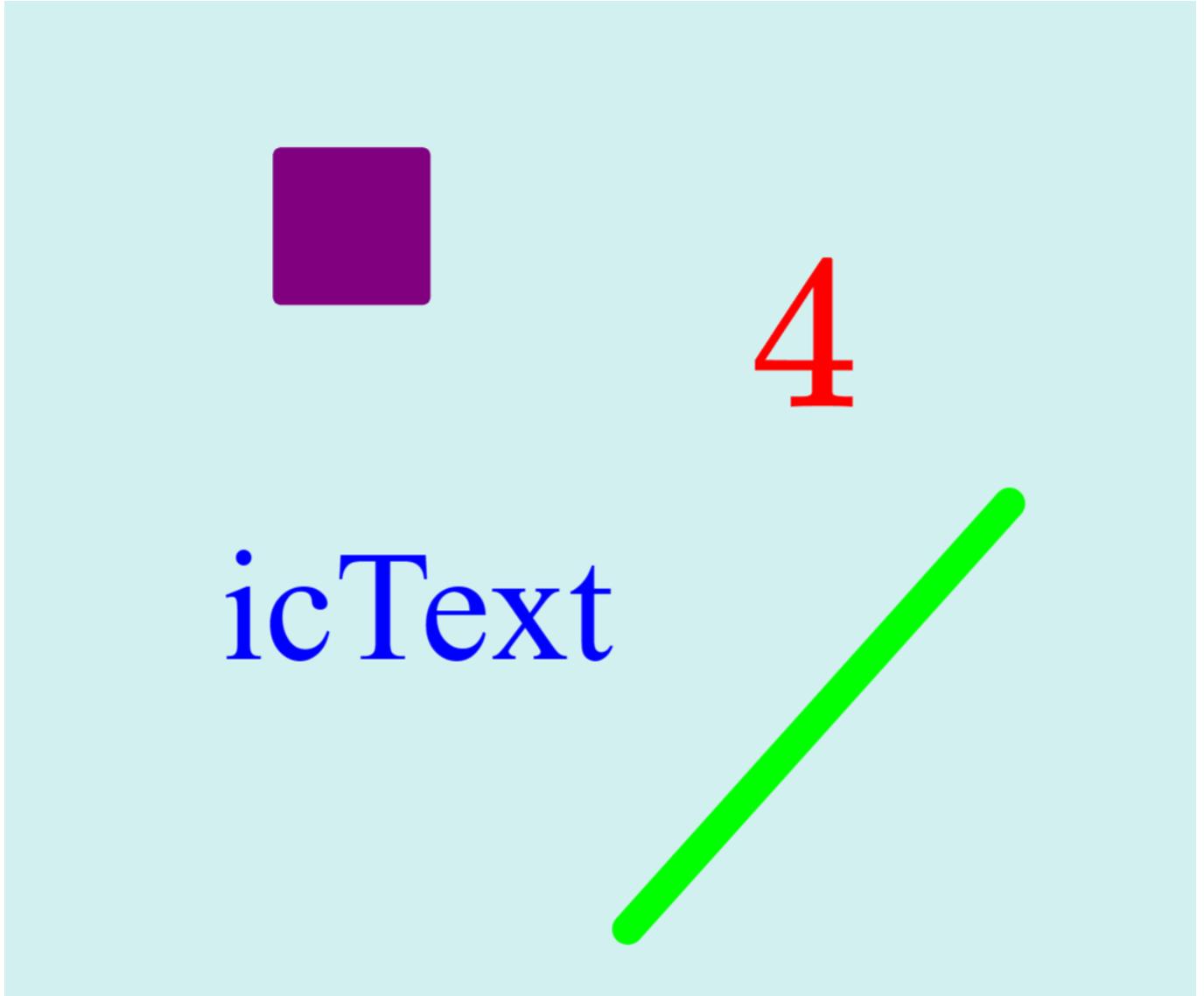
Les concepts ont été nommés dans une sorte d'anglais qui permet de former des mots en patchwork, compacts et assez clairs. C'est assez normal étant donné que le développement de logiciel en programmation textuelle demande la création et l'emploi de mots d'une sorte d'anglais-machine. Mais pas de panique, c'est un livre en français, avec plein de dessins, pour des humains.

Il y a des zones prospectives avec des verbes au conditionnel : Eh oui, c'est plus rapide d'écrire une idée que d'écrire le code qui la réalise.

Aller, c'est parti !

I. Première partie: InfiniteCanvas (iC)

0. icObj, icComposite, engine



tout est objet

Dans Misocroft PoperWoint, il y a des zones de texte, des images, etc. Dans iC, c'est la même chose, sauf qu'il y a aussi des objets mathématiques. L'exemple type est iCMathNode, qui pour faire simple, représente un nombre. Autrement dit, comme dans un langage de programmation classique, on fait la différence entre les nombres et les chaînes de caractères. iCImage est une image. iCText est du texte. iCRect est un rectangle. iCSegment est un segment. Toutes les sortes d'objets peuvent exister dans iC (il y a même iCIInfiniteCanvas...) On peut saisir un objet et le déplacer en cliquant dessus et en glissant la souris tout en restant cliqué. On peut faire défiler l'écran de façon panoramique (pan), en restant appuyé sur le clic droit de la souris. On peut combiner déplacement d'un objet et défilement panoramique pour emmener un objet vers une zone préalablement non-apparente.

infiniteCanvas

Jusque là, c'est comme dans PoperWoint. Toute la différence vient maintenant: il y a de la place pour tout le monde : puisque l'espace est infini, quelle que soit la taille de l'objet que l'on veut insérer, on pourra toujours

le faire. Dans un canevas infini, la zone qui est affichée à l'écran est un point de vue. On regarde juste un morceau d'un espace 2D infini. On peut zoomer et voir des objets qui étaient tellement minuscules qu'ils n'occupaient même pas un pixel sur l'écran de l'ordinateur. On peut dézoomer et voir que l'on était en fait "à l'intérieur" d'un objet très grand.

Objets motorisés:

Les objets présentés jusque là, les nombres, les images, les textes, n'ont pas de comportements autonomes. Ils sont juste des conteneurs de données. On peut les qualifier de "minéraux".

Les moteurs permettent d'ajouter de la vie aux objets "minéraux". Exemples: Avec un moteur de bouton, on peut transformer un objet texte, en un bouton, de telle sorte qu'à chaque fois que l'on va cliquer dessus, on va ajouter le caractère '.' à la fin. On passera donc par exemple de "abc" à "abc." puis "abc.." puis "abc..." etc. On peut donner un battement de coeur à un nombre: chaque seconde, il change de couleur. Un même objet peut avoir plusieurs moteurs (tant qu'ils sont compatibles).

Composites:

Un composite est un conteneur d'objet. C'est à dire qu'un composite peut contenir d'autres objets, qui peuvent être eux-aussi des composites, ... On peut facilement s'imaginer cela avec des cartons: dans un carton, on peut mettre des objets "normaux" (ballon de basket, banane, rouleau de scotch), et d'autres cartons (avec des choses dedans). Les composites qui contiennent un objet sont ses composites parents (il y a sa mère, sa grand-mère, etc.). Le composite qui n'est contenu dans rien est le composite racine. En pratique, un objet composite est un rectangle (icRect) équipé d'un moteur "composite". Un composite peut avoir des parties mobiles, c'est à dire des parties que l'on peut déplacer librement par rapport aux autres, et des parties fixes, qui entraînent le déplacement "en bloc" de tout le composite racine. Ces objets sont très utiles pour créer des structures (structures de données mais également dispositions visuelles). Exemples: On prend un nom, un prénom, un âge, une photo et on l'on crée une structure "Elève". Lorsque l'on déplace le composite élève, nom, prénom, âge et photo se déplacent en même temps. Autrement dit, c'est un objet en soi, de la même manière qu'une voiture, avec ses passagers et son chargement, peut être considérée comme un seul objet.

1. propriétés des objets

Position généralisée:

Pour retrouver une information posée sur le canevas, il faut connaître sa position et son niveau de zoom (échelle familière (homeScale)). On pourrait appeler ça sa "position généralisée".

Objets avec échelle, objets sans échelle, objets avec et sans échelle:

Objets avec échelle:

Les objets standard ont une position (x,y) sur le canevas, une taille(largeur, hauteur) mais également une échelle familière. Exemple: une image quand on dézoomé beaucoup, on ne la voit plus. quand on zoomé beaucoup, l'image prend tout l'écran et on peut zoomer sur des détails de l'image(bien que ça devienne flou à partir d'un moment) Si on se place à l'échelle qu'a l'image, on la voit bien nette. C'est le niveau de zoom idéal pour la regarder.

Objets sans échelle:

L'exemple typique est le point. Quel que soit le niveau de zoom auquel on se place, il est affiché toujours de la même taille à l'écran. C'est un objet sans échelle. En zoomant suffisamment, on peut connaître sa position avec une précision arbitraire. C'est la façon que l'on a de représenter le point, qui n'a pas de dimension.

Objet avec et sans échelle:

L'affichage du point dans les logiciels de géométrie dynamique est parfois envahissant. Quand on est très dézoomé, le point est toujours à l'écran, ce qui pollue l'affichage dans certains cas. L'idée est de fixer une échelle, à partir de laquelle, le point se comportera comme un objet standard, c'est-à-dire qu'il deviendra plus petit, jusqu'à ce qu'on ne le voit plus du tout. Puis quand on zoomera à nouveau dessus, il grossira pour rejoindre sa taille de point et son comportement d'objet sans échelle. On appelle ces objets des objets à échelle mixte (mixedScale objects).

zIndex**objets spéciaux:**

- fantôme

Cet objet n'apparaît que lorsque l'utilisateur se place à un niveau de zoom proche de son niveau de zoom familial. Quand on est trop zoomé ou trop dézoomé, il n'est pas visible.

- façade

Cet objet disparaît quand l'utilisateur s'est placé à un niveau de zoom trop important. Cela donne l'impression de traverser la façade et donne accès à ce qu'il y a derrière.

- autres:

Certains objets changent de comportement suivant le niveau de zoom auquel on se place. Par exemple: de loin, la fenêtre d'une maison apparaît fermée et elle s'ouvre quand on s'approche.

2. zScroll, et tous les autres scrolls

De la bonne utilisation de la molette de souris

Quand la molette de souris est apparue dans les années 1990, c'était, originellement, pour faciliter la navigation des comptables et des gestionnaires sur des feuilles de calcul avec beaucoup de cellules, via des zooms et des dézooms. Malheureusement, cette vision n'a pas vraiment été réalisée. La molette a surtout servi à faire défiler ("scroller") verticalement et horizontalement. Il a fallu attendre l'apparition des cartes virtuelles pour que la molette retrouve enfin sa mission principale: zoomer et dézoomer.

Par la suite, on va supposer que l'utilisateur est droitier.

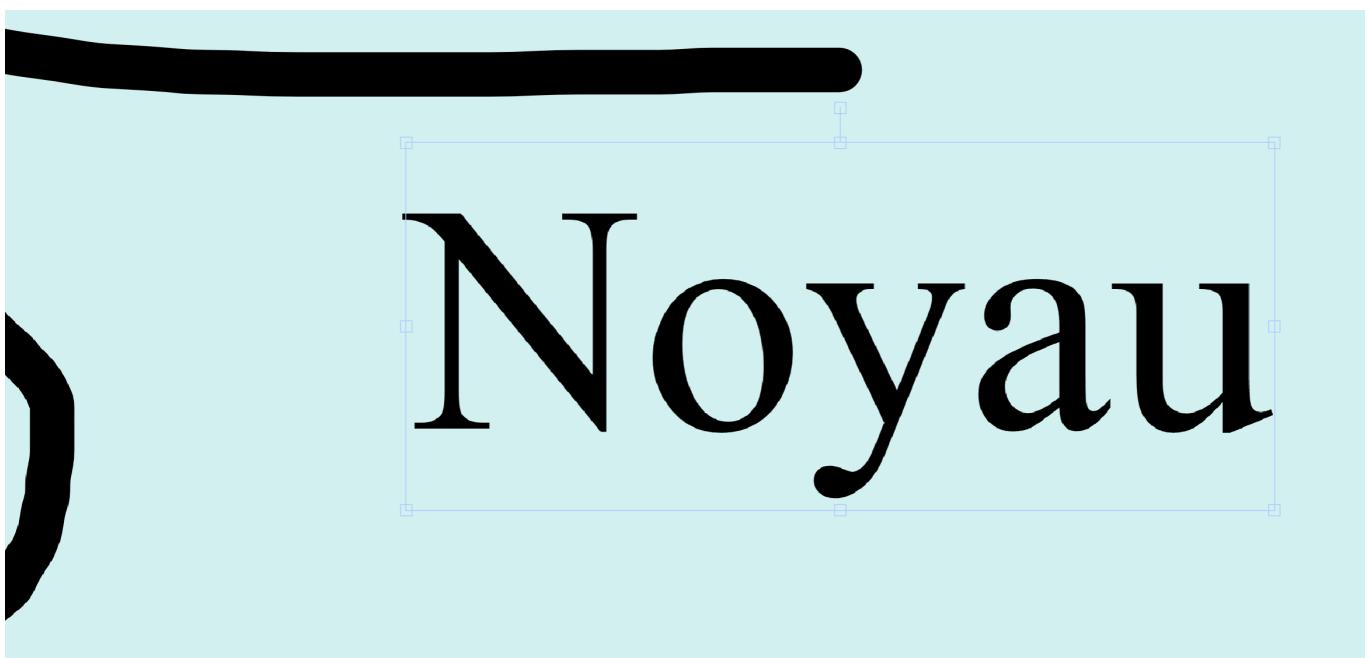
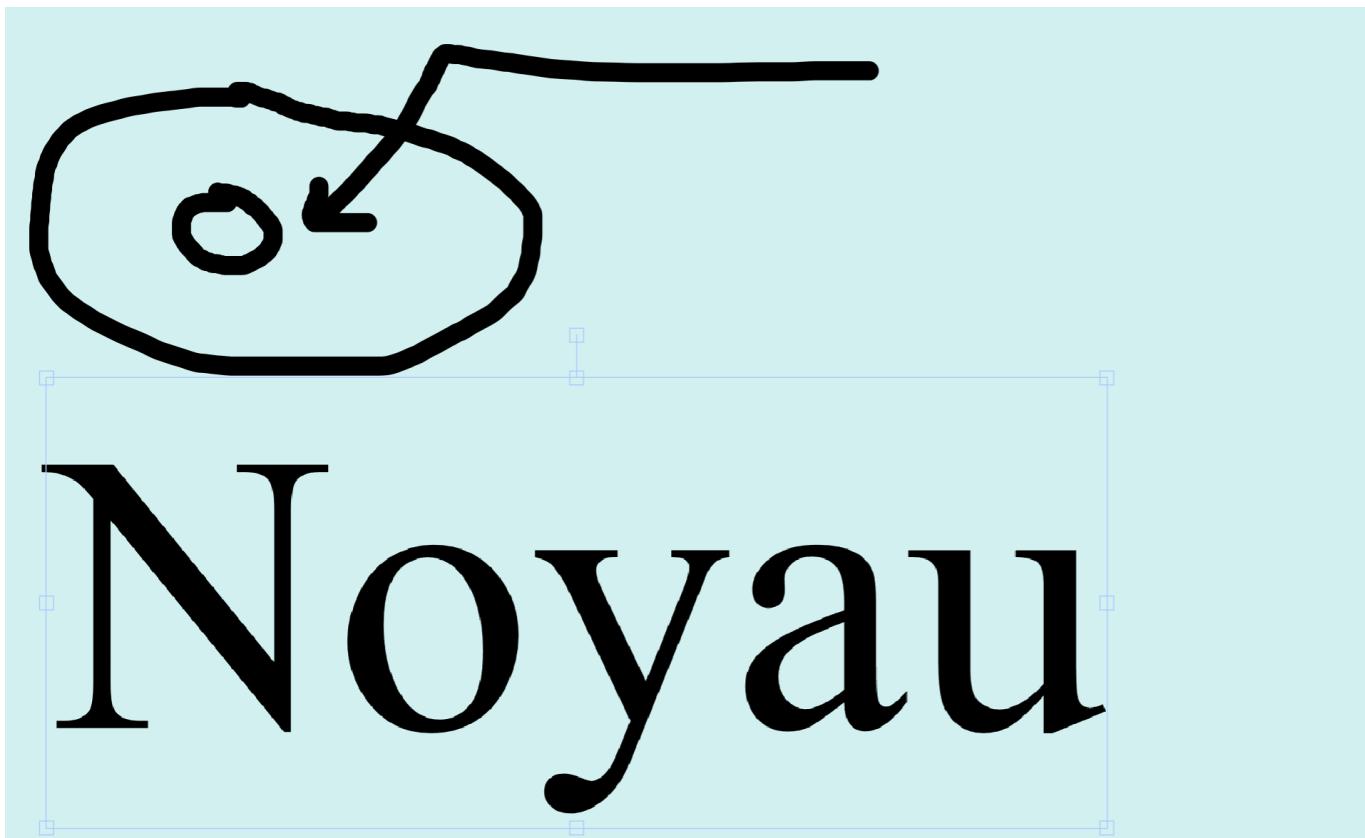
L'interface utilisateur est optimale, quand la souris est utilisée de manière optimale. Pour une utilisation optimale, il est préférable d'utiliser la molette avec l'index. Il s'en suit qu'il est préférable d'interchanger le clic droit et le clic gauche.

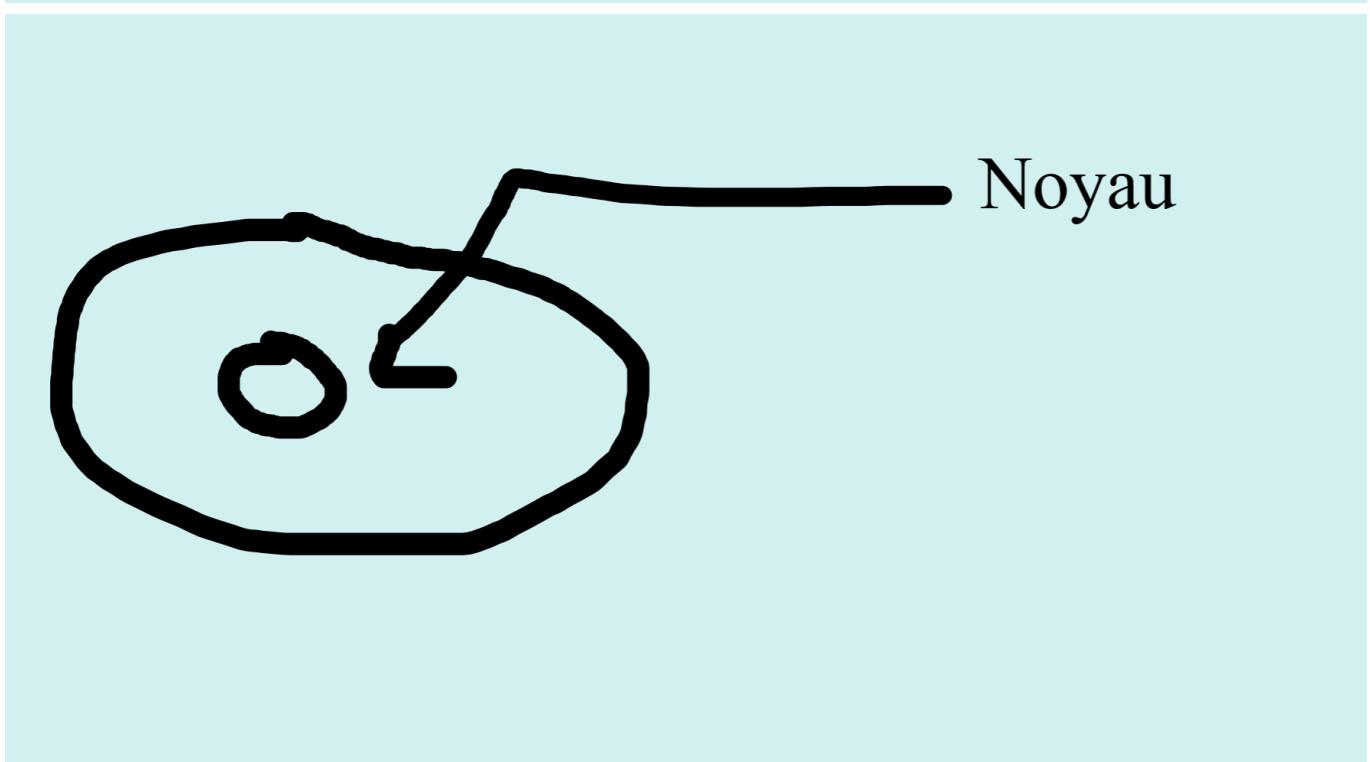
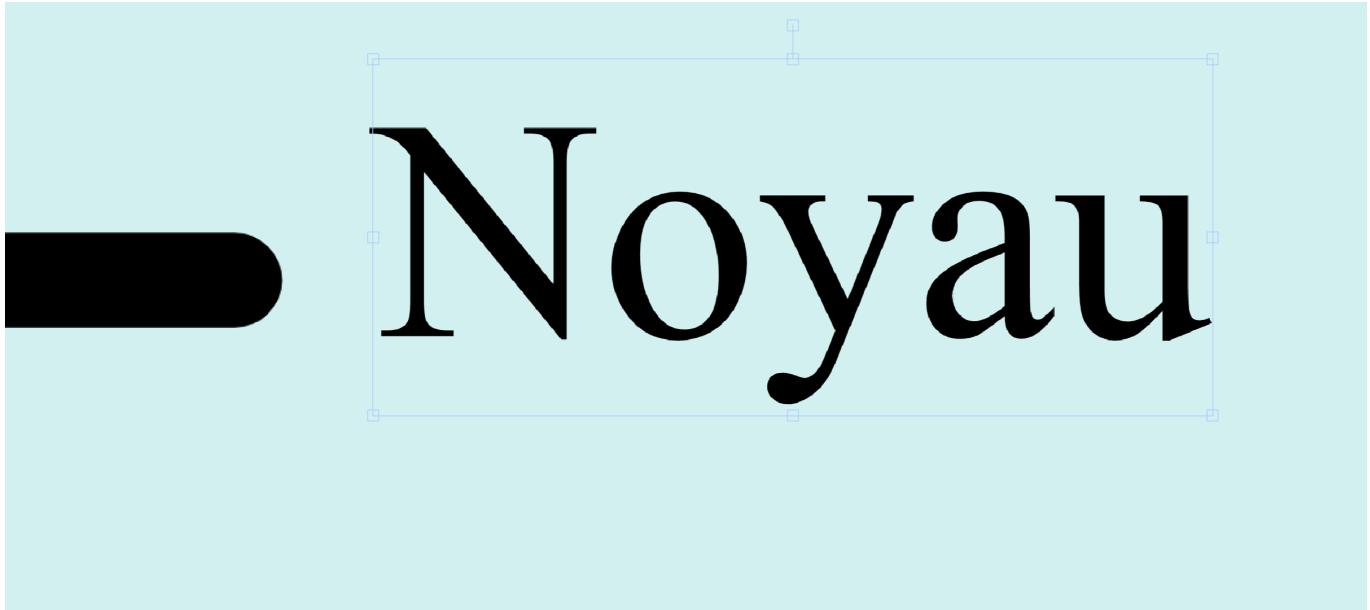


Par la suite, on continuera d'appeler "clic gauche" le clic standard (qui se situera maintenant à droite sur la souris). "Clic droit" désignera le clic alternatif (on le fera avec l'index, sur le bouton gauche de la souris).

zScroll (le retour d'une vieille idée)

Pour être tout à fait précis, la vision initiale pour la molette de souris n'était pas seulement de zoomer et dézoomer. C'était zoomer et dézoomer *pour déplacer* une valeur d'une cellule à une autre sur la feuille de calcul (Eric Michelman en 1993). Autrement dit, ne pas seulement changer le point de vue, mais faire voyager de la donnée. Le "zScroll" c'est cette idée, appliquée au canevas infini: On attrape un objet, et, alors qu'on zoomé et qu'on dézoomé, cet objet reste avec nous. Le zScroll est tellement facile à réaliser (on clique, on roule la molette), que l'utilisateur l'utilise rapidement de façon naturelle. L'utilisateur prend également conscience que la taille n'a plus vraiment d'importance, puisque tout ce qui est de taille finie peut être ramené à une taille "opérable", en utilisant le zScroll.





C'est une idée maîtresse, qui rend l'utilisation du canevas infini très très confortable. Pour augmenter la vitesse de zoom/dézoom, on peut rester appuyé sur la touche Shift du clavier (touche souvent associée au sprint dans les jeux vidéo), et effectuer le mouvement de zScroll classique. On n'est pas obligé de garder saisi l'objet. En fait, il suffit qu'il soit sélectionné. Le zScroll change alors seulement l'échelle de l'objet et pas sa position.

les autres scrolls

Une fois que l'on a compris à quel point le zScroll était agréable, on a envie de tirer le meilleur parti de la molette de souris. En appuyant sur une touche du clavier, on peut indiquer que l'on souhaite faire autre chose que le comportement standard, le zScroll. En voici quelques uns.

valueScroll (touche N + molette)

On saisit un objet nombre, on appuie sur la touche N du clavier et on scrolle: la valeur est incrémentée. Pour faire simple, si la valeur était 4, elle devient 5. Si on continue de scroller vers le haut, elle devient 6, etc. Si on

scrolle vers le bas, la valeur est diminuée. C'est une façon rapide de créer le nombre que l'on souhaite.

opacityScroll (touche V + molette)

On saisit un objet, on appuye sur la touche V du clavier et on scrolle: son opacité est modifiée. On peut facilement créer des objets "un peu transparents".

Je parle fort.

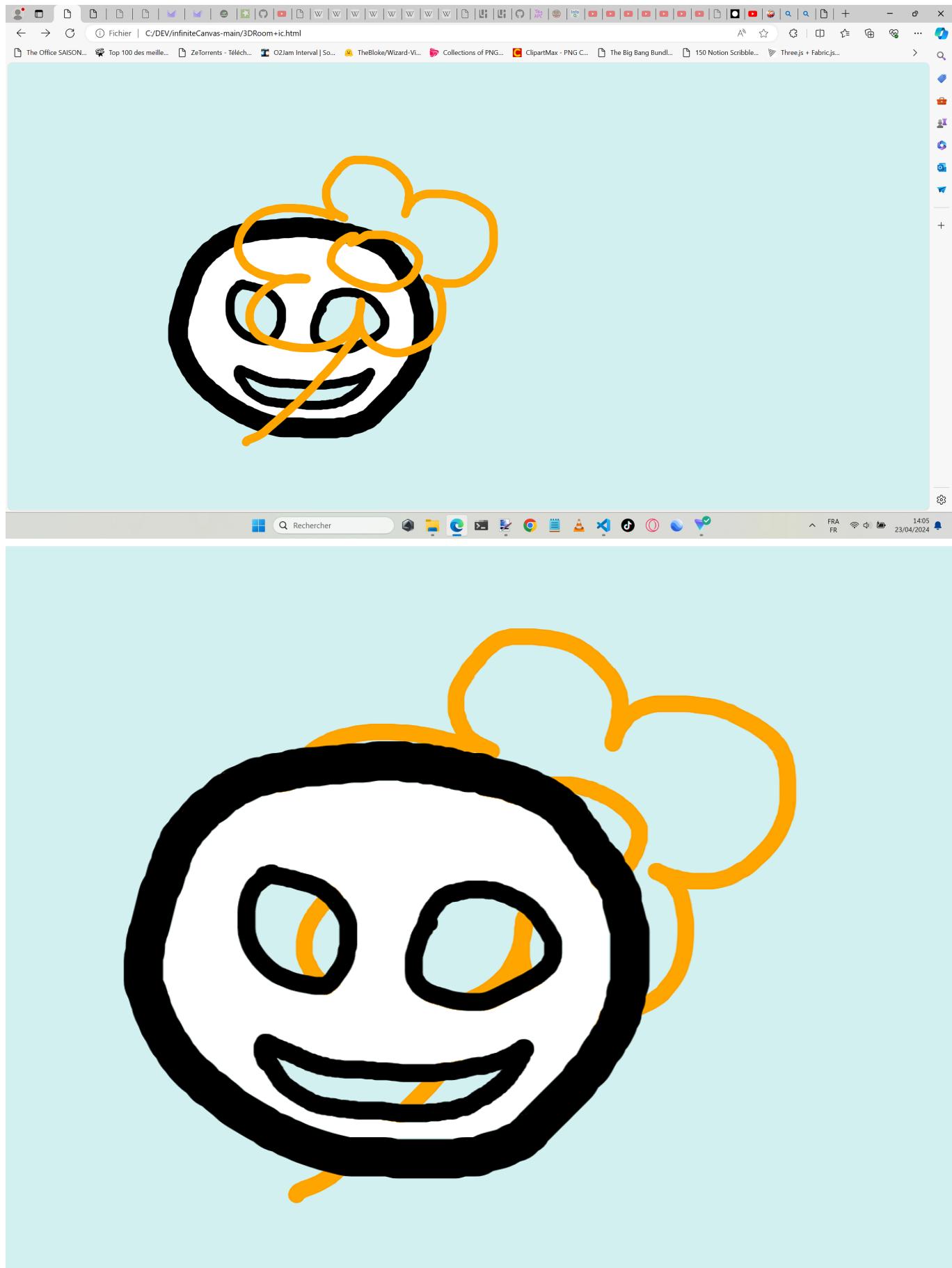
Je parle.

Je murmure.

colorScroll: on change la couleur de l'objet

Cette fonctionnalité est très agréable car elle permet de changer la couleur principale d'un objet très rapidement, sans avoir à passer par un menu.

zIndexScroll: on fait passer l'objet au premier plan, ou un peu derrière un autre objet, ou tout au fond, etc. (on change sa disposition)



3. opérateurs, formalisme, commandes

Il n'y a pas de menu : la transformation des objets se fait sur place. Il y a une catégorie d'objets dédiée aux transformations, on les appelle les opérateurs.

opérateurs standard:

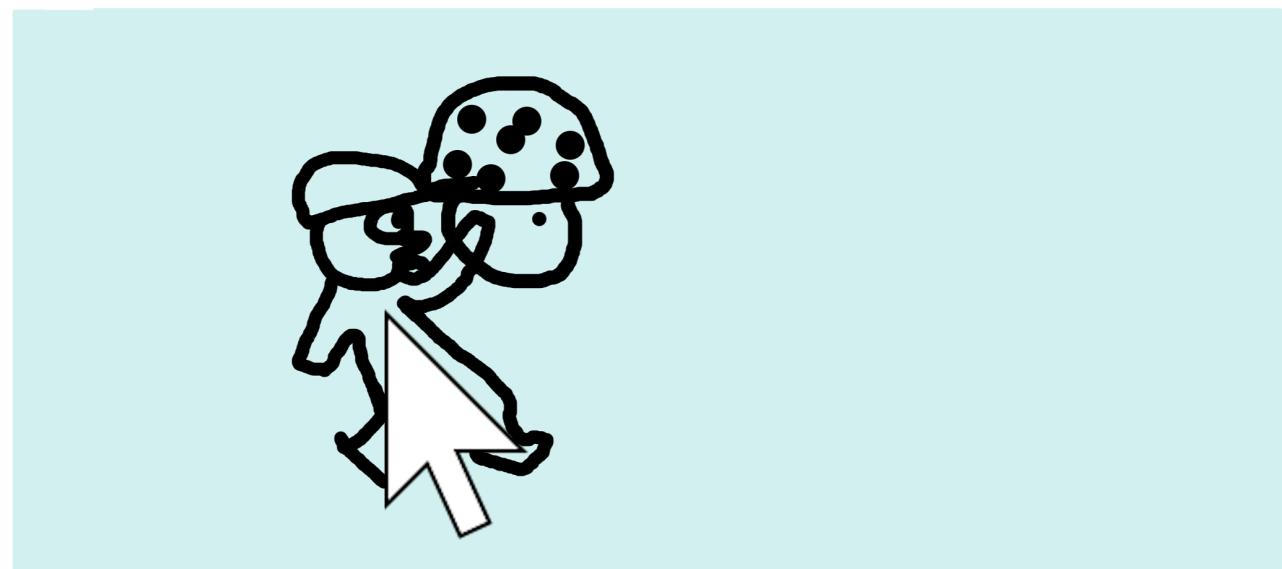
Pour transformer un objet:

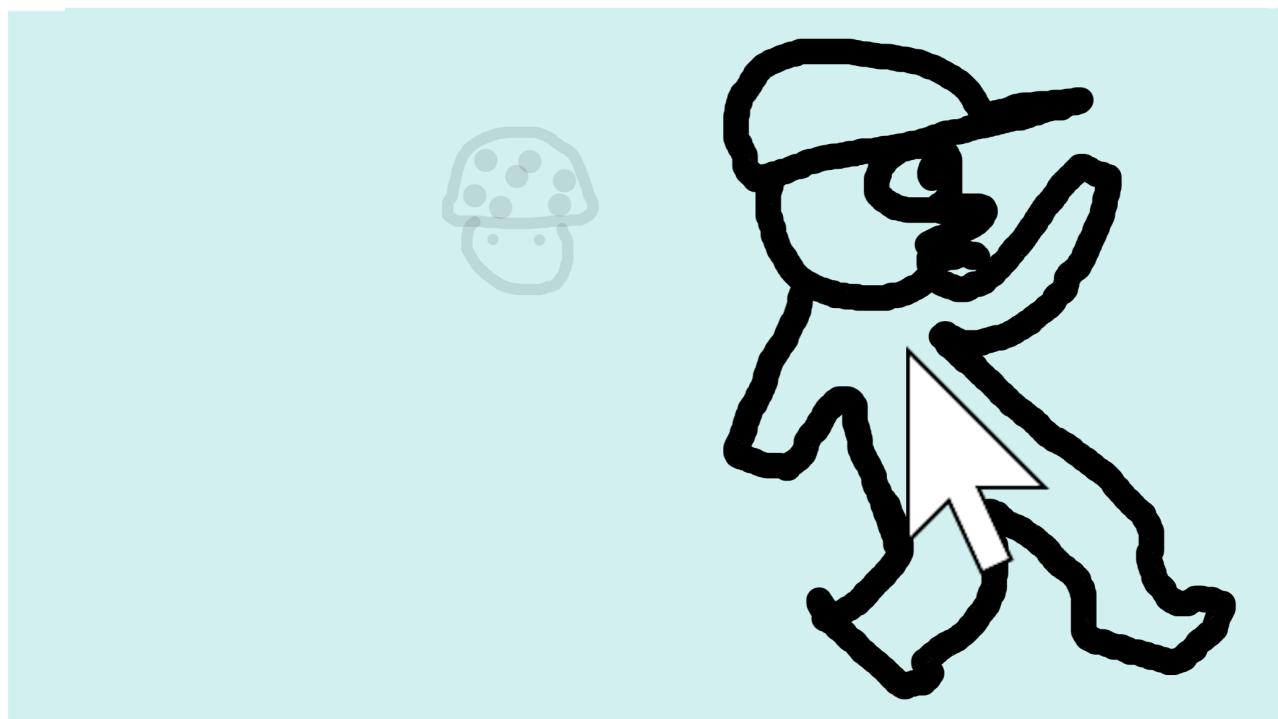
- on se saisit de l'objet,
- on lui fait toucher un opérateur et l'opérateur s'applique alors à l'objet.

Tout simplement.

En fait, c'est une idée à laquelle sont habitués beaucoup d'enfants:

Quand Mario touche le champignon, il devient grand-Mario.





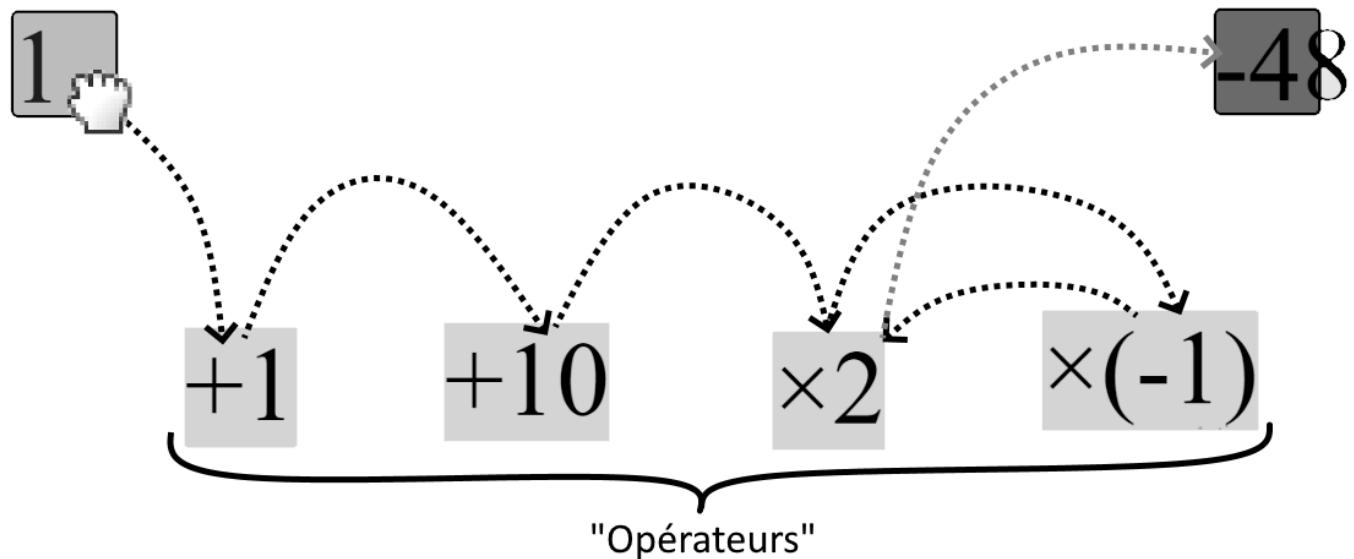
Tout ce qui change, c'est qu'on l'attrape avec la souris, et que le champignon n'est pas détruit une fois utilisé.

Dans Logo de Seymour Papert, il y a la tortue d'un côté et la programmation textuelle de l'autre.

Dans Scratch, il y a le chat d'un côté et la programmation par blocs de l'autre.

Dans Une Interface Utilisateur (AUI), on efface la barrière entre le chat et le code.

Le chat, c'est l'objet que l'on transforme, et le code, ce sont les opérateurs qu'il va rencontrer. Dans le processus, le chat est devenue un peu du code (il contient de la donnée), et le code est devenu un peu un chat (il a une position, une taille, une couleur...). Code is Data and Data is Code. (D'ailleurs on peut anticiper que les opérateurs peuvent eux aussi être transformés par d'autres opérateurs)



Formalisme basique:

Pour parler de l'interface utilisateur avec du texte, on symbolise les différents objets. On utilise des symboles enveloppants car les objets sont avant tout des conteneurs.

Les objets sont indiqués par des doubles crochets.

`[[4]]` désigne l'icMathNode de valeur 4.
`[["abc123"]]` désigne l'icText contenant la chaîne de caractères "abc123"

Les opérateurs sont indiqués par des simples crochets.

`[+ 2]` désigne l'opérateur qui ajoute 2.
`[EnMajuscules]` désigne l'opérateur qui met en majuscules le texte en entrée.

L'interaction entre un objet et un opérateur est indiquée par le symbole '*' et le résultat vient après le symbole '='

`[[4]] * [+ 2] = [[6]]`
`[["abc123"]] * [EnMajuscules] = ["ABC123"]`

C'est une réaction de chimie virtuelle,

(* et = correspondent respectivement à + et → dans une réaction chimique).

```
[[Mario(petit)]] * [champignon] = [[Mario(grand)]]
```

opérateurs applicables:

Il ne se passe rien quand on touche un opérateur applicable avec un objet à transformer. Pour l'actionner, pendant que l'objet saisi est touché par l'opérateur, on fait un clic droit. Ces opérateurs sont symbolisés entre parenthèses. Exemple: (+ 2) Ces opérateurs ajoutent une petite couche de protection par rapport aux opérateurs classiques.

Comment créer des opérateurs ?

Une première façon de faire est d'entrer en mode "Saisie de Nombre"(en appuyant sur la touche W du clavier) et de taper, par exemple, "+ 2".

```
Attention, "+2" fera apparaître l'objet nombre [[2]]  
alors que "+ 2" fera apparaître l'opérateur [+ 2].  
Saisir "x|->3x+2" fera apparaître [x->3x+2].
```

Remarque en lien avec la programmation:

Dans un but pédagogique, et pour insister sur les transformations, la valeur que retourne la fonction est directement assignée à l'objet en entrée. L'objet entrant subi une mutation. Il est tout à fait envisageable, dans une évolution ultérieure, et pour former un public spécifique en informatique, de réaliser des fonctions pures pour lesquelles l'objet d'entrée et l'objet de sortie seront bien distincts, et où les affectations se feront uniquement en utilisant "[←]".

générateurs:

Il y a des opérateurs applicables très utiles: les générateurs.

```
En mode saisie, "(|->2)" fera apparaître (↔2).  
Cet opérateur s'utilise de la manière suivante:  
on s'en saisit, et à chaque fois que l'on fait un clic droit, un objet nombre  
[[2]] apparaît dans le canevas.
```

Les générateurs sont particulièrement utiles pour générer des nombres aléatoires.

```
(↔0:1) donnera [[0]] avec 50% de chance et [[1]] avec 50% de chance.  
(↔1:2:3:4:5:6) est le D6 classique.
```

Les commandes ou comment ne pas créer des opérateurs

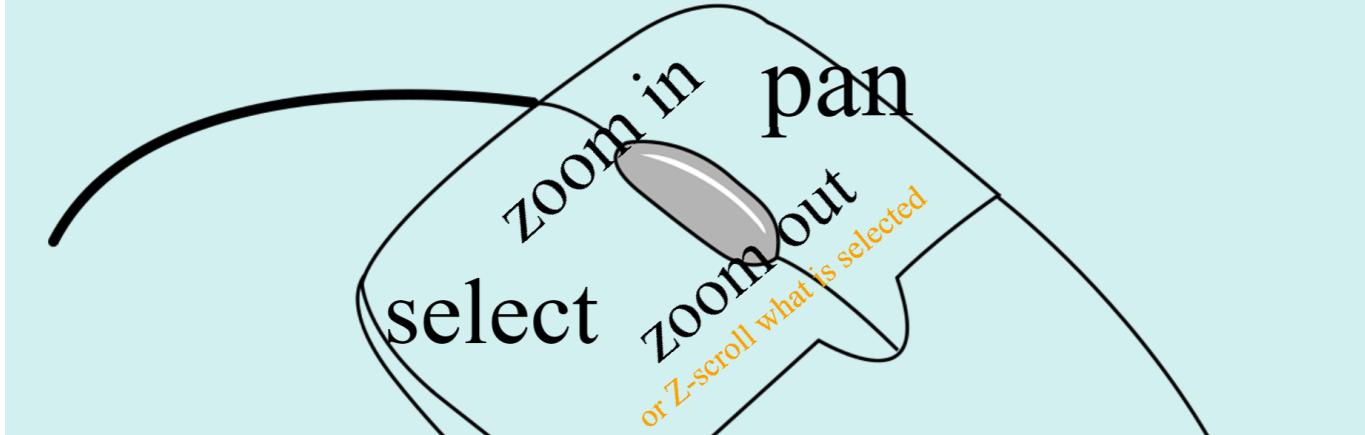
Certaines opérations sont si courantes qu'il est plus confortable pour l'utilisateur de les invoquer sans passer par un opérateur. L'utilisateur utilise une commande, symbolisée par !! Il y a un raccourci clavier/souris pour invoquer la commande.

!save!	Touche S	enregistrer le contenu du canevas
!load!	Touche L	charger du contenu sauvegardé
!import!	Touche I	importer une image, une vidéo, un fichier pdf, un fichier epub,...
!deselect!	Touche Espace	Désélectionner l'objet ou le groupe d'objets saisi
!clear!	Touche K	effacer tout le contenu du canevas
!clone!	Touche X	Faire apparaître un clone de l'objet sélectionné, à l'endroit où l'objet est actuellement
!spawn_rect!	Touche F	Faire apparaître un rectangle ayant la couleur courante
!colorScroll!	Touche C + molette	Changer la couleur courante
!numberTyping!	Touche W	Créer un nombre, une expression ou un opérateur
!freeDrawing!	Touche ²	Mode dessin

Forcément, la maîtrise des commandes passe par une phase d'apprentissage, un peu comme pour la guitare, ou un nouveau smartphone. Mais, alors que les raccourcis clavier pourront éventuellement changer, pour se conformer à chaque type d'usage, à l'initiative de l'utilisateur, les concepts de base seront difficiles à désapprendre. Le vélo change mais ça reste un vélo.

Le tutoriel, un contenu comme un autre:

Une Interface Utilisateur (AUI) a son tutoriel, contenu dans le canevas infini. Pas besoin de manuel externe au logiciel. Les fonctionnalités associées à chaque touche du clavier et de la souris sont inscrites directement sur celles-ci.

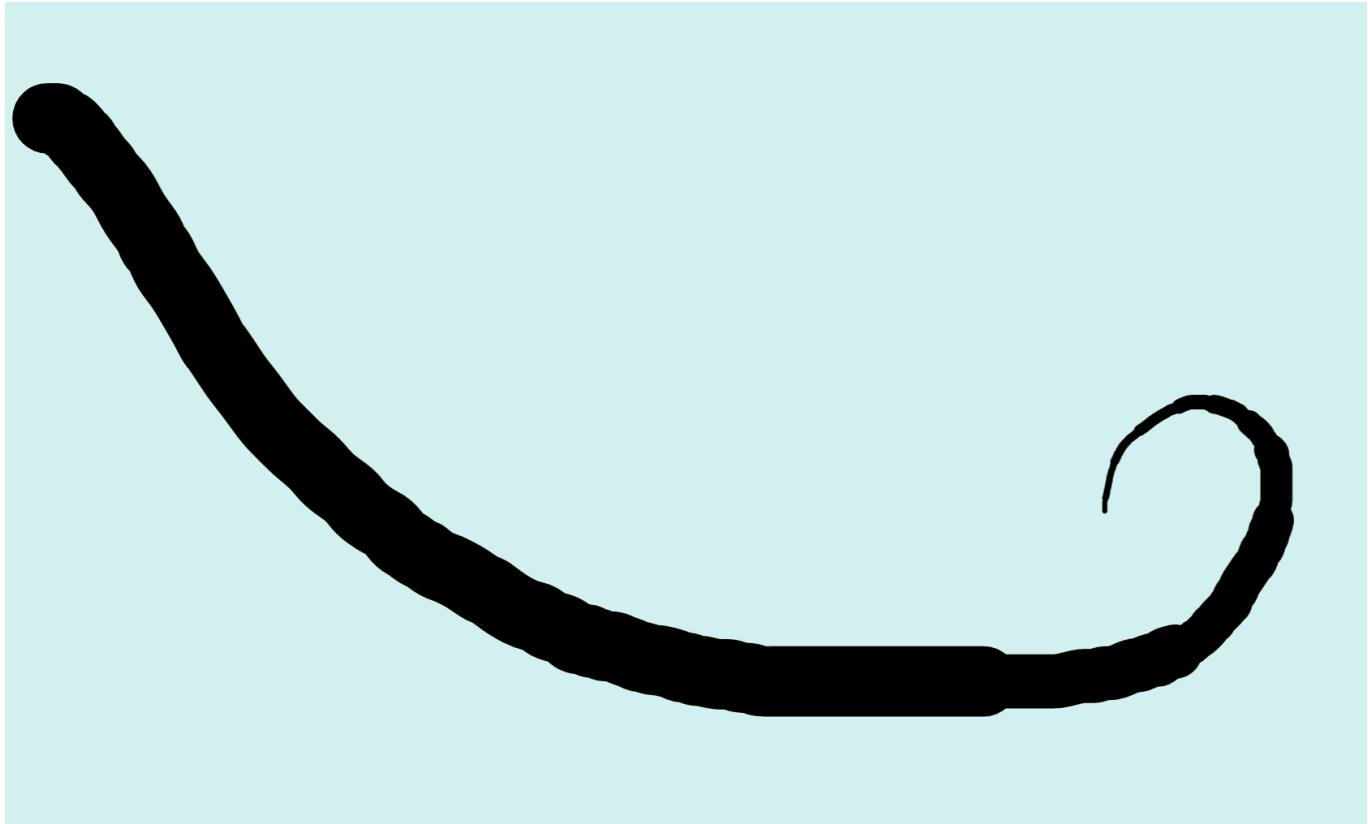


Et on peut facilement imaginer des tutoriels spécifiques à chaque fonctionnalité, en zoomant encore dans la touche concernée, avec des exercices. Le niveau de maîtrise de chaque fonctionnalité serait affiché: une touche blanche correspondrait à "aucune maîtrise", une touche orange correspondrait à "en cours de maîtrise", une touche verte à "bonne maîtrise". En regardant le clavier de loin, on visualiserait, en un coup d'œil, où l'on en est.

4. freeDraw: adhere, comb, ..., vDraw

freeDrawingZscroll:

Les fonctionnalités de l'interface utilisateur ont été parfois découvertes comme tentative de mariage de deux concepts existants. (Une "matrice d'innovation" pourrait d'ailleurs être utilisée pour essayer d'épuiser de manière exhaustive les ouvertures possibles.) Ici, on mélange le dessin à main levée classique avec le zScroll. Il en résulte un dessin à main levée dont le trait suit le niveau de zoom.



Deux conséquences directes, assez difficile à faire sur du papier physique :

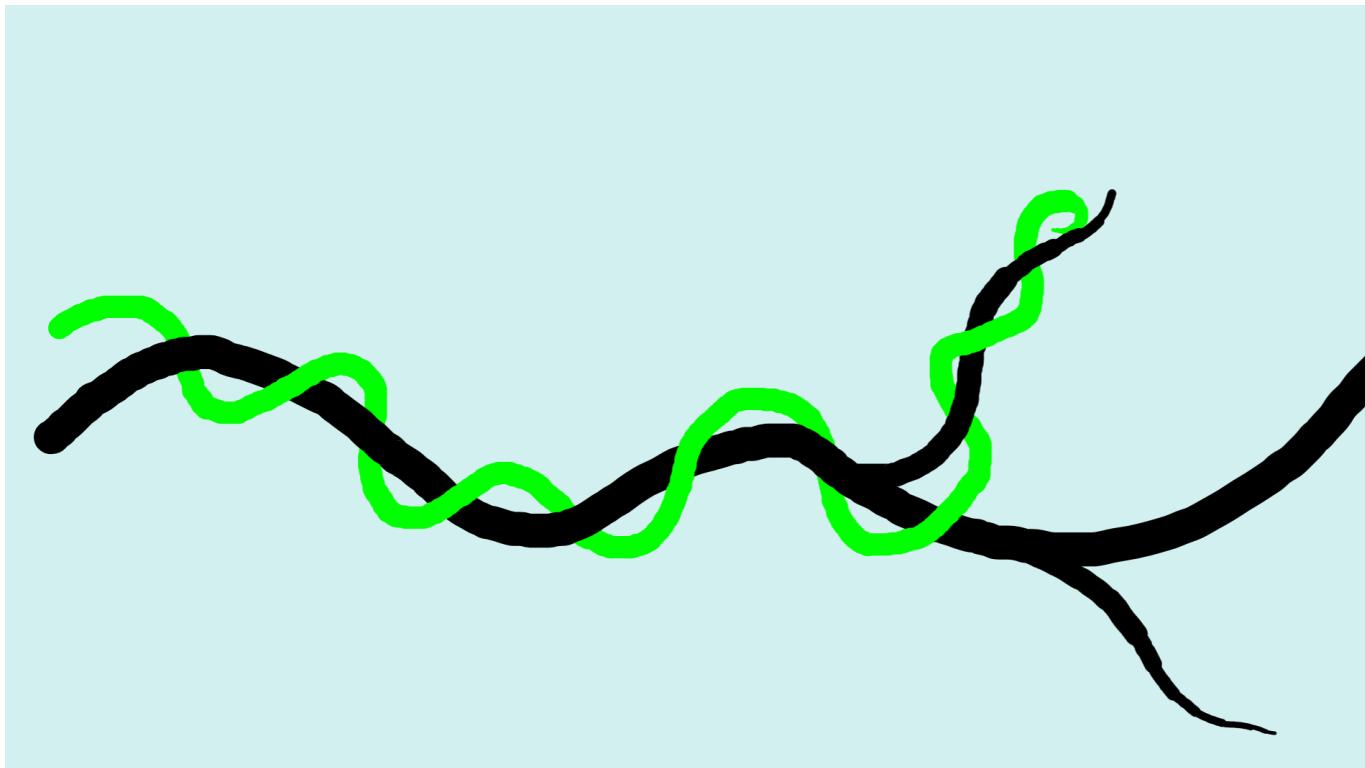
- On peut dessiner très simplement des traits à l'épaisseur variable. On peut faire des poils, des spirales...
- On peut créer des dessins pour lesquels l'échelle est une dimension, au même titre que "le x et le y". On peut mettre autant de détails que l'on souhaite dans l'arrière plan. On peut avoir un arrière-plan de l'arrière-plan, etc. puisque "arrière-plan" est simplement une appellation, qui dépend de l'endroit où l'on est placé.

**brushWidthScroll:**

Comme le freeDrawingZscroll, mais sans suivre le zoom. On change simplement l'épaisseur du trait, en même temps qu'on le trace.

freeDrawingZIndexscroll:

Cette fonctionnalité est inspirée des graffiti. On change la disposition (la "frontalité") du trait à mesure que l'on dessine. Conséquences directes, pratiquement impossibles à faire sur du papier physique : On peut dessiner un trait qui s'enroule comme un serpent autour d'un autre trait. On peut dessiner un trait qui fait comme un fil de couture sur une face de tissu. On peut dessiner des fourreaux semi-transparents dans lesquels on peut fourrer des objets ou desquels on peut dégainer des objets.

**adhere:**

Quand un enfant dessine, il découpe parfois le personnage qu'il vient de dessiner car il le considère comme un seul objet, alors même qu'il lui a fallu plusieurs traits pour le dessiner. L'adhérence réalise cela dans iC :

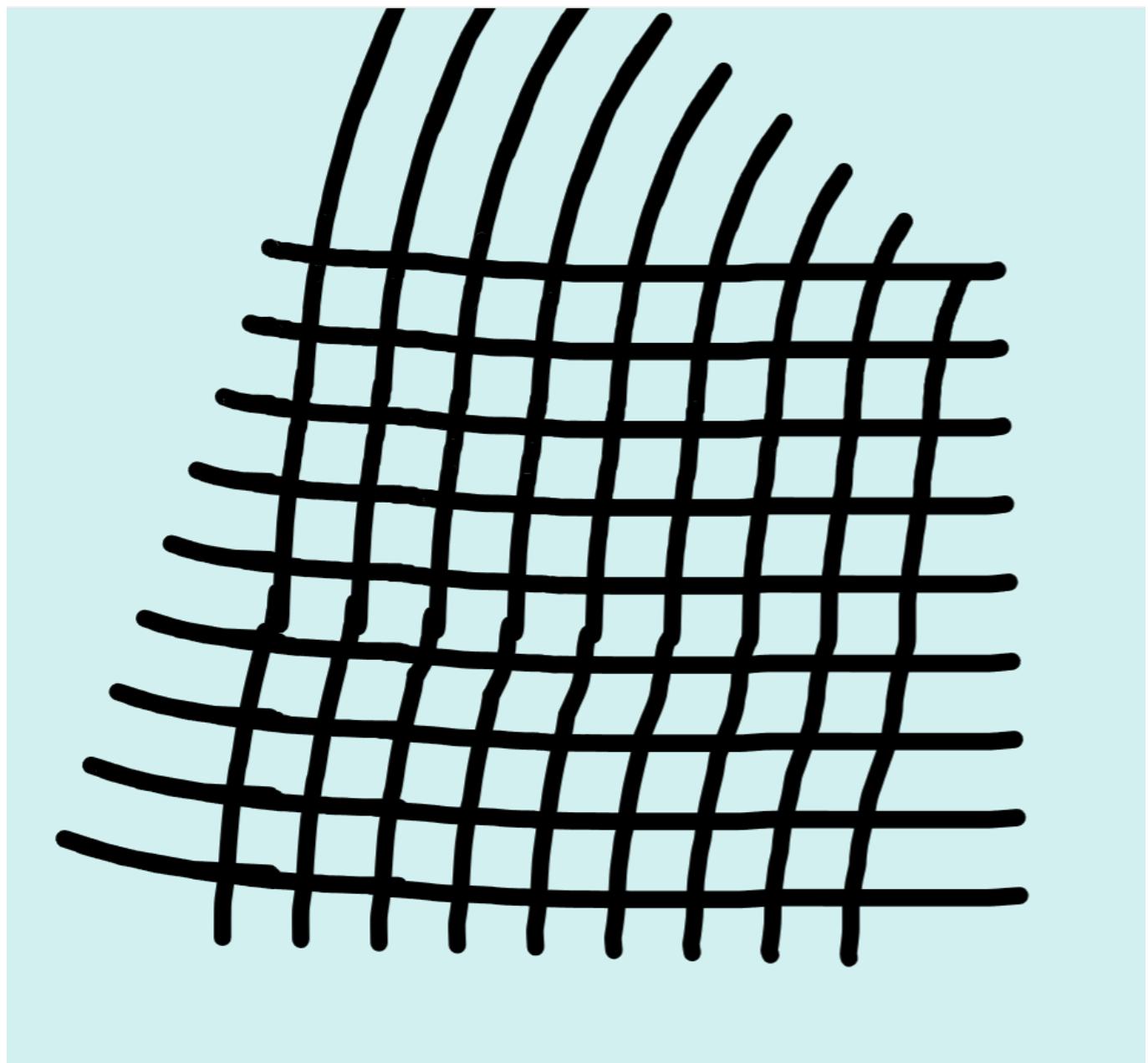
quand on trace un trait en commençant à tracer sur un trait déjà existant, le trait "du dessus" adhère sur le "trait support" et les deux forment un composite.

On peut alors déplacer les deux traits "en bloc".

Cela permet de constituer un dessin complexe, de manière constructive, mais sans séparer l'acte de dessiner et l'acte de structurer.

comb:

Cette fonctionnalité est inspirée de la peine que l'on a quand on trace des tableaux à la règle et au stylo. L'idée est que si l'on avait un stylo au bout de chaque doigt de sa main, en un passage vertical et un passage horizontal, on pourrait avoir un tableau 4x4. On choisit le nombre de traceurs que l'on veut utiliser et les traceurs suivent le curseur de manière naturelle.

**combScroll:**

Quand on a 2 traceurs ou plus, le combScroll permet de contrôler l'écartement des traits. Il faut s'imaginer dessiner à deux doigts et, tout en les avançant, les écarter et les rapprocher pour former des pincements et des bourrelets. On peut s'amuser à tracer des circuits auto, en refermant le parcours sur lui-même.

**vDraw:**

Que se passe-t-il si on dessine avec autre chose que le stylo ? Et si on dessinait avec un objet ? Le vDraw consiste à dessiner avec un nombre, pour indiquer facilement ce que l'on est en train d'en tracer une représentation à main levée.

Je dessine un 2 avec [[2]].

Je pourrais alors utiliser ce 2 à main levée comme si c'était un [[2]] standard.

Je dessine deux points avec [[2]].

Je pourrais alors utiliser ces deux points comme si c'était un [[2]] standard.

police facile:

On peut dessiner des adhérences sur les caractères d'un texte tapé au clavier. On modifie le graphisme de chaque caractère très facilement. Le système comprend que l'on crée une nouvelle police de caractère.

On fait des oreilles et un oeil au caractère 'a'.

La prochaine fois que l'on tapera le caractère 'a', il apparaîtra de la même manière que le caractère modifié.

Un utilisateur peut créer une police de caractère facilement de cette façon.

Avant le monde était à nous, maintenant il est à personne

5. opérateurs cheminants (pathmade): vers la programmation par le dessin (codessiner)

On sait faire des opérateurs, on sait dessiner, et si on mélangeait les deux ? De la même façon que l'on peut créer des objets complexes en utilisant les composites, on peut créer des opérateurs complexes en utilisant des tapis roulants . Voici le principe :

en dessinant un tapis roulant dont le trajet passe par des opérateurs, on arrive à transformer l'objet de départ selon une séquence bien précise.
Il y a aussi des opérateurs cheminants que l'on crée en reliant des objets, ou en saisissant des valeurs en même temps que l'on dessine.

Appelons-les les opérateurs cheminants. Dans la logique Logo → Scratch → Une Interface Utilisateur (AUI), les opérateurs cheminants permettent de créer des enchaînements d'instructions, à l'instar des empilements de blocs dans Scratch.

Les illustrations vont parler d'elles-mêmes.

Structures de contrôle: if on dessine des branchements switch on dessine la colonne puis on dessine des branchements il y a un branchement pour "default:" boucle on dessine une boucle pour faire une boucle clonage on dessine des branchements sans conditions Les programmes deviennent des objets comme les autres, dans le canevas infini.

sleeping-args ops:

Une façon assez naturelle d'utiliser des opérateurs qui prennent plus qu'un argument en entrée, est de poser les arguments sur l'opérateur en attendant d'aller chercher les autres. Une fois que l'opérateur a suffisamment d'arguments en entrée pour s'activer, le résultat est retourné. Les arguments qui attendent sont appelés arguments dormants (sleeping args en anglais). L'ordre des arguments est donné par la position qu'ils occupent sur l'opérateur. On peut aussi utiliser des convoyeurs d'arguments pour rendre l'ordre plus explicite.

tDraw:

Que se passe t'il quand on décide de dessiner une fonction, non pas avec le pinceau "spécial fonction", mais avec un objet type ? On dessine une fonction typée, c'est-à-dire une fonction dont on prévoit à l'avance le type des arguments. On peut spécifier de cette manière les types d'arguments en entrée et les types de résultats en sortie. On peut d'ailleurs imaginer que des programmes dessinés soient compilés et transformés

en opérateurs standard pour avoir des exécutions aussi rapides que possibles, une fois le développement terminé.

Débuggeur:

La vitesse de convoyage des objets sera réglable pour bien comprendre les exécutions, ce qui sera utile pour enseigner la programmation (ralentir la vitesse), et pour enseigner les mathématiques (augmenter la vitesse pour faire des statistiques autour de processus aléatoires, ou pour tracer des graphes de fonctions point par point).

Autres développements:

fonctions avec plongements Si on dessine des chemins qui plongent (pour une explication, voir "freeDrawingZscroll" juste après) et qui remontent ensuite, on peut représenter le fait qu'on "descende d'un cran" quand on rentre dans une fonction. On peut mettre une façade, comme un capot sur le moteur, pour cacher le fonctionnement interne.

!record!:

Cette commande permet d'entrer en mode enregistrement Toutes les actions que l'on effectue sont enregistrées et on peut alors les rejouer à l'identique. On enregistre les actions et pas les états, ce qui fait que l'on peut appliquer de manière répétée un processus de transformation à un objet.

On enregistre la séquence $[[2]] * [+ a] * [+ b] = [[2+a+b]]$
et on replace $[[2+a+b]]$ là où était $[[2]]$ quand l'enregistrement a commencé.
Si on la rejoue, on aura:
 $[[2+a+b]] * [+ a] * [+ b] = [[2+a+b+a+b]]$
etc.

La programmation par le dessin semble être une piste intéressante pour commencer l'enseignement de la programmation. Une progression de la forme:

programmation par le dessin → programmation par blocs → programmation textuelle

pourrait amener les élèves en douceur jusqu'à la programmation textuelle.

La programmation par le dessin pourrait également servir au professeur à écrire rapidement des programmes destinés à la classe: générateurs aléatoires, programmes de tri, ...

Application lointaine: aller vers l'électronique en simulant des cartes et des composants, en modélisant les impulsions électriques par des objets et les fils par des convoyeurs.

Fonctions induites: ouverture vers l'intelligence artificielle

6. points de vue: povRect/printPovRect/screenPovRect/portails/icPeer

Un morceau rectangulaire du canevas infini -aussi petit soit il- est encore infini, car on peut zoomer indéfiniment à l'intérieur. En fait, on peut se dire que le rectangle que l'on voit sur l'écran n'est qu'une certaine prise de vue sur le canevas infini.

povRect:

Plus simplement, on peut retrouver un objet en faisant coïncider le cadre de l'écran avec un rectangle qui le contient, de manière "assez proche". C'est-à-dire que le rectangle conteneur doit avoir une échelle proche de celle de l'objet. On va appeler ça un povRect (rectangle point-de-vue). L'utilisateur se balade dans le canevas, trouve un point de vue qui lui plaît et le matérialise : tout est objet et le povRect n'échappe pas à la règle, comme [[4]] ou [[["Salut"]]]. Cela permet à l'utilisateur de le déplacer facilement. L'utilisateur peut retourner à un point de vue particulier en appuyant sur le bouton qui y est associé. Il peut bien sûr mettre ce bouton n'importe où. De manière astucieuse, en mettant à côté d'un point de vue le bouton vers le point de vue suivant. on peut créer des enchaînements comme les diapositives dans Misocroft Poperwoint, mais les transitions seront des glissements avec zooms/dézooms. On peut même ajouter des informations qui seront affichées uniquement pendant les transitions.

printPovRect:

Les povRects sont utilisés pour faire des présentations dynamiques, ou pour se déplacer dans le canevas. Si on se place dans un contexte éducatif, avec l'envie d'imprimer du cours ou des exercices pour les élèves, on peut utiliser les printPovRects. C'est le même principe mais ici on va utiliser des formats de page d'impression. Exemple: un printPovRect format A4, avec le numéro #1, correspondra à la première page du document à imprimer. Ceci permettra de ne pas avoir de mauvaises surprises à l'impression car on précise que ce que l'on voit est ce que l'on veut. Le document final peut être prévisualisé dans le canevas en rassemblant de manière horizontale, toutes les pages capturées. L'objet document est lui aussi un objet du canevas, et donne d'autres possibilités d'édition (changer l'ordre des pages par exemple). Une version canevas infini du livre est disponible en chargeant (Touche L) le fichier aui_book_fr.shu depuis 3dRoom+ic.html

screenPovRect:

le point de vue, correspondant à ce qui est affiché à l'écran, et qui se déplace sans cesse, est appelé screenPovRect. Ce que voit ce point de vue peut être rediffusé dans un autre objet : comme un écran qui diffuserait ce qu'une caméra filme.

portails: voyager, la tête sur la feuille

Le jeu vidéo Portal a permis aux joueurs de vivre ce qui était par exemple décrit par Dan Simmons dans Hyperion, passer d'un endroit de l'univers à un autre en traversant simplement une porte. Sans surprise, dans le canevas infini, on peut faire la même chose. Il y a des objets appelés portails qui, quand on les traverse, comme si on plongeait dedans, nous transportent vers un autre povRect du canevas. Il peut donc se passer les voyages suivants: plongeon: on plonge dans un portail et on se retrouve dans une zone "très profonde" retour en arrière: on plonge dans un portail et on se retrouve dans une zone qui contient le portail que l'on a pris Exemple: Le portail dans l'oeil d'une mouche nous fait nous retrouver de la taille de la tour Eiffel, la mouche étant posée sur un CupriSan, lâchement abandonné sur le sol du 1er étage, au milieu des touristes. dépaysement total: on plonge dans un portail et la nouvelle zone n'est ni conteneur ni contenu du portail source. Exemple: on plonge dans un élément d'une page de livre pour se retrouver quelque part sur la page

d'à côté. Portails aller-retour: le franchissement du portail "dans l'autre sens" nous fait revenir où on était avant. Remarque: un portail peut-être utilisé pour cacher de l'information. Tout ce qui est sous le portail d'entrée n'est plus visible, car ce qu'on l'on voit à l'intérieur, c'est ce que contient le portail de sortie.

iC in iC: hierarchiser mais sans abandonner l'infini

iCCanvas Le canevas infini est un élément html qui couvre toute la page, c'est donc un rectangle qui ne change jamais de taille. Cela étant dit, on peut imaginer créer un objet iCCanvas, qui contient lui aussi un canevas infini. On peut entrer dedans et naviguer à l'intérieur. Pour en sortir, soit on utilise un portail à l'intérieur de celui-ci pour nous ramener au canevas parent, soit on utilise la commande !iC_out! (comme dans le 'kick' dans le film Inception, on remonte au niveau de rêve supérieur). Ceci s'avèrera très utile pour la coopération, car le canevas d'un utilisateur invité pourra être intégré au canevas de l'hôte, de la même façon. iC "troué": encore plus de place Si l'on s'arrange pour que le canevas fils contienne la zone du canevas parent qu'il masque par sa présence, alors on a encore plus de place. Le canevas fils est alors une sorte de sas de taille infinie. opérateurs cheminants intergalactiques: La création d'un nouvel objet invite à repenser tous les objets existants, tous les opérateurs, toutes les commandes, etc. Si l'on a des canevas infinis dans des canevas infinis, on peut avoir des convoyeurs traversant les canevas, traversant les portails. Il est difficile d'imaginer ce que cela pourrait donner sans avoir commencé à développer de tels objets. De nombreux studios de développement de jeux vidéo ont déjà de l'expérience dans le domaine, en 3D, donc cela semble tout à fait faisable en 2D. A l'avenir, créer un canevas fils sera peut-être aussi naturel que de créer un dossier dans un explorateur de fichier.

icPeer: tirer le meilleur parti des réseaux locaux

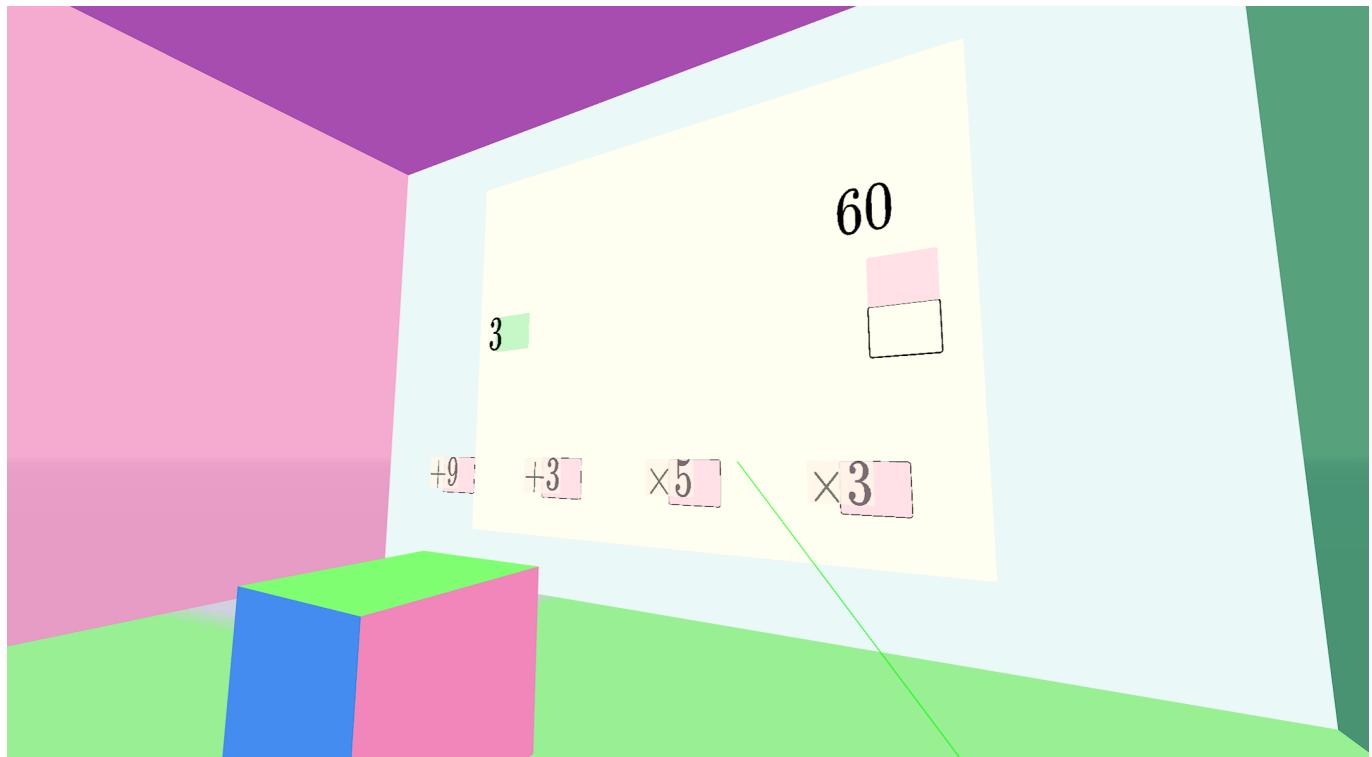
Douglas Engelbart a eu un impact énorme sur l'informatique. En 1968, dans "The mother of all demos", il a montré à l'humanité qu'il était déjà dans le futur. On lui doit bien sûr la souris d'ordinateur et cette volonté de tirer le meilleur parti des ordinateurs, avec des interfaces utilisateurs puissantes, pour travailler seul et mais aussi, et cela l'intéressait beaucoup, en coopération. Une Interface Utilisateur (AUI) ne fait que continuer les pistes qu'il a ouvertes, avec une attention particulière pour les interfaces élève-machine. Voilà une situation de classe typique: les élèves sont en salle informatique, chacun devant un ordinateur, et le professeur se demande pourquoi, (2024-1968=)56 ans après l'impact du météore Engelbart sur la Terre, il est pratiquement impossible de les faire travailler ensemble, en binôme par exemple, sur un même projet numérique. Notre utilisation des ordinateurs en classe est loin d'être optimale, et les enfants le savent, puisqu'ils sont habitués aux interfaces des jeux vidéos et des réseaux sociaux: il est bien plus facile pour eux de jouer en réseau ou de chatter sur leur téléphone qu'en classe. Le canevas infini pourrait apporter juste assez de simplicité d'utilisation pour résoudre ce problème.

Une fois connecté sur sa session d'élève (réseau local de son établissement ou réseau local de la salle informatique), et "devant" son canevas infini, il y verrait alors des objets représentant les canevas de ses camarades. A partir de ces objets, l'élève pourrait faire une demande de connection avec un camarade, et pourra, une fois la demande acceptée, zoomer dans la vignette de son camarade pour naviguer dans son espace, et par exemple, amener avec lui (zScroll) un objet pour lui donner, être conduit par son camarade jusqu'à une zone particulière, etc. Ils pourraient aussi avoir un canevas infini dédié à leur coopération, qui n'appartiendrait ni à l'un ni à l'autre. Accès franc: un accès franc est l'utilisation d'un transport d'un client vers un autre. IMAGE Le web: Et pour une utilisation sur Internet, une liaison de pair à pair serait optimale : Quand on travaille avec quelqu'un, on n'a pas besoin du reste de la planète. Contrairement au confort que le réseau local apportait aux élèves, en générant automatiquement les objets de connexion entre élèves, l'utilisateur devra créer l'objet de connexion (icPeer) avec l'utilisateur avec qui il veut entrer en connection (par exemple

en scannant un QRCode). Mais une fois tout ces détails réglés, ce serait comme en classe dans la salle numérique. On pourrait travailler partout d'une façon unifiée. Il faut simplement passer d'un état d'esprit "Malheureusement c'est impossible" à un état d'esprit "Il n'y a qu'à le faire", autrement dit retrouver l'état d'esprit des pionniers de l'informatique.

7. De l'infini partout: iC+3dRoom/revisiter les affichages classiques

iC est un canevas infini 2D. 3dRoom est la même chose en 3D. 3dRoom peut contenir des iC.



Conséquence: On peut se balader librement dans cet espace 3D et s'arrêter pour explorer un espace infini. L'affichage peut être capturé pour avoir une vraie sensation 2D, où alors, comme une simulation, afficher un tableau comme le voit un élève depuis son bureau dans une salle de classe. De la même manière qu'un iC peut contenir des iC, une 3dRoom peut contenir des 3dRoom. Cela rend l'espace beaucoup plus grand que la façon dont on a l'habitude de l'appréhender.

L'exploration de ce genre d'espace est comme une chasse aux oeufs psychédélique, dans laquelle il y aurait des oeufs planète, des oeufs bactérie, des oeufs galaxie, des oeufs nucléon, etc. Chacun peut avoir son petit jardin secret sur un pétalement de tournesol, ou cacher son journal intime dans le point final d'une phrase.

zLivres:

De la même manière que l'on a changé le scroll habituel de défilement vertical en scroll de zoom, on peut commencer à imaginer de nouvelles formes d'affichage de l'information.

Sandy mangeait tranquillement son jambon-beurre.

Sandy

C'est la meilleure copine de Lily.



C'est la meilleure copine
de Lily.

Un livre, au format électronique, c'est une grande colonne de texte. C'est linéaire. On peut tout à fait imaginer des "zLivres", exploitant pleinement la dimension "échelle" du canevas infini. Beaucoup d'artistes ont déjà expérimenté les dessins en "z2D" (il y a même des publicités en "zoom,zoom,zoom,zoom,zoom").

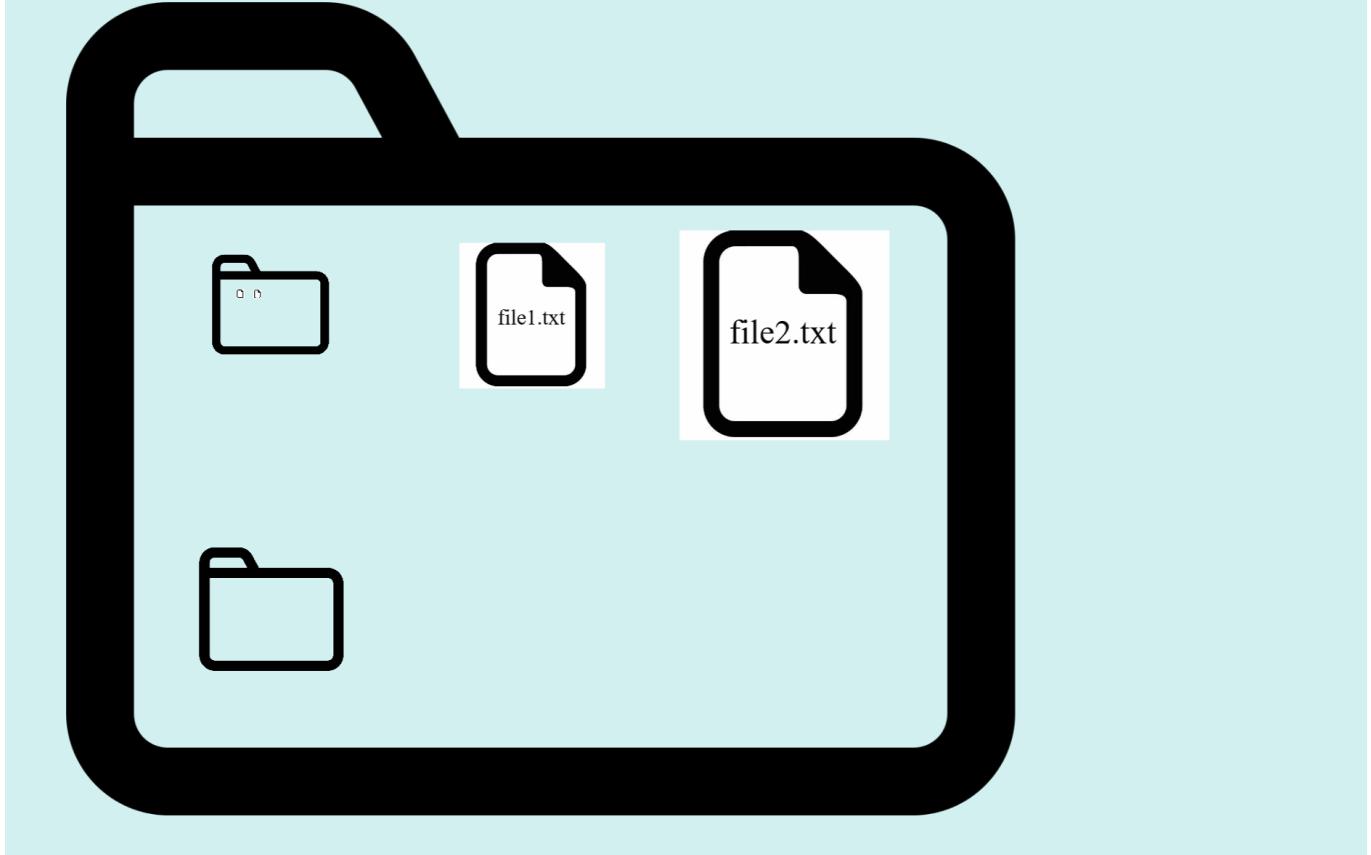
zWeb:

Le web pourrait être transformé ainsi, avec une navigation en zoom et dézoom, en plongeons dans les portails. Après une période de transition où se cotoieraient vieux style et nouveau style, (imaginer prendre un

article sur un site marchant et le déposer dans le panier, ou créer un portail vers une zone de jeu ou un article pour le partager à un ami). Il pourrait y avoir un espace web territorial dans lequel les sites des acteurs locaux d'un territoire seraient affichés: le site du boulanger, le blog de mon voisin, le site de l'association de basket, ... Plus on dézoomerait, plus on passerait à une échelle nationale, puis internationale. Il pourrait y avoir un espace web sémantique où les pages reliées à une page wikipedia lui seraient adjacentes.

zOS:

Pour faire très simple, un système d'exploitation sert principalement à gérer ses fichiers et à exécuter des programmes. Les fichiers seraient représentés par des objets "fichier", les programmes par des opérateurs. La métaphore du dossier serait revisitée: on zoomerait dans un dossier pour voir son contenu.



Mais le sujet est très vaste et bien plus complexe.

II. AUI pour l'éducation (Green Mouse (gm)/edu)

8. Une souris verte: gm/touchSlot

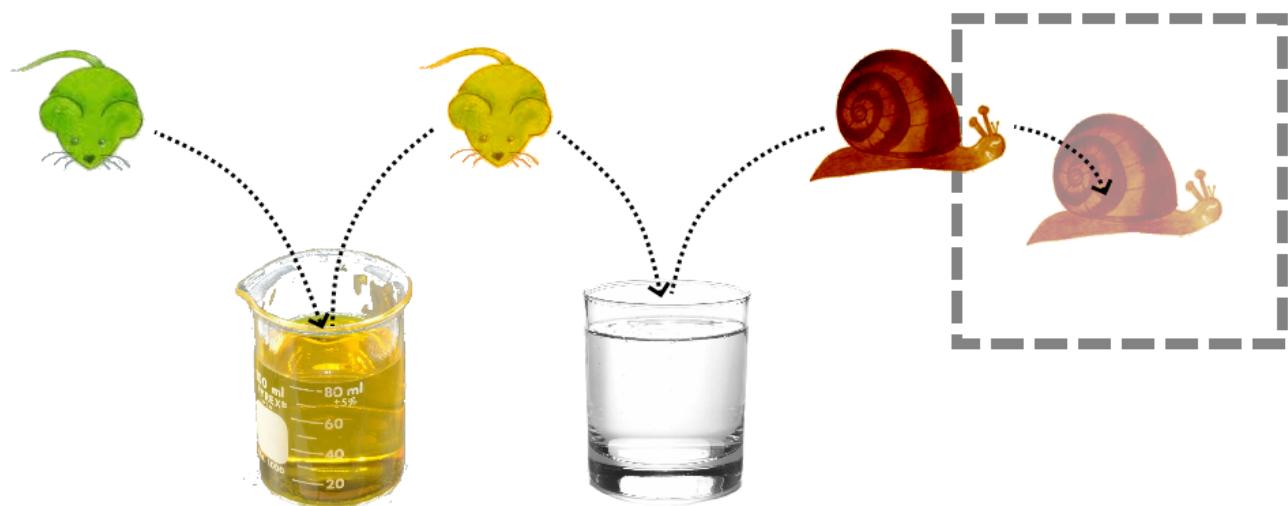
les malédictions

Les objets du canevas peuvent avoir des malédictions. Exemples de malédictions: A chaque fois que l'objet est touché par un opérateur, il change de couleur. Si l'objet est touché plus de 5 fois par un opérateur, il est détruit.

La souris verte(greenMouse, abbrévié en gm):

Appliquer la transformation directement quand l'objet touche l'opérateur est rapide et confortable. Les opérateurs standards se comportent comme des cuves dans lesquelles on trempe l'objet saisi. L'objet

transformé ainsi fait penser à la comptine de la souris verte. Une souris verte, qui courait dans l'herbe, je l'attrape par la queue, je la montre à ces messieurs. Ces messieurs me disent Trempez-la dans l'huile, Trempez-la dans l'eau, ça fera un escargot tout chaud.

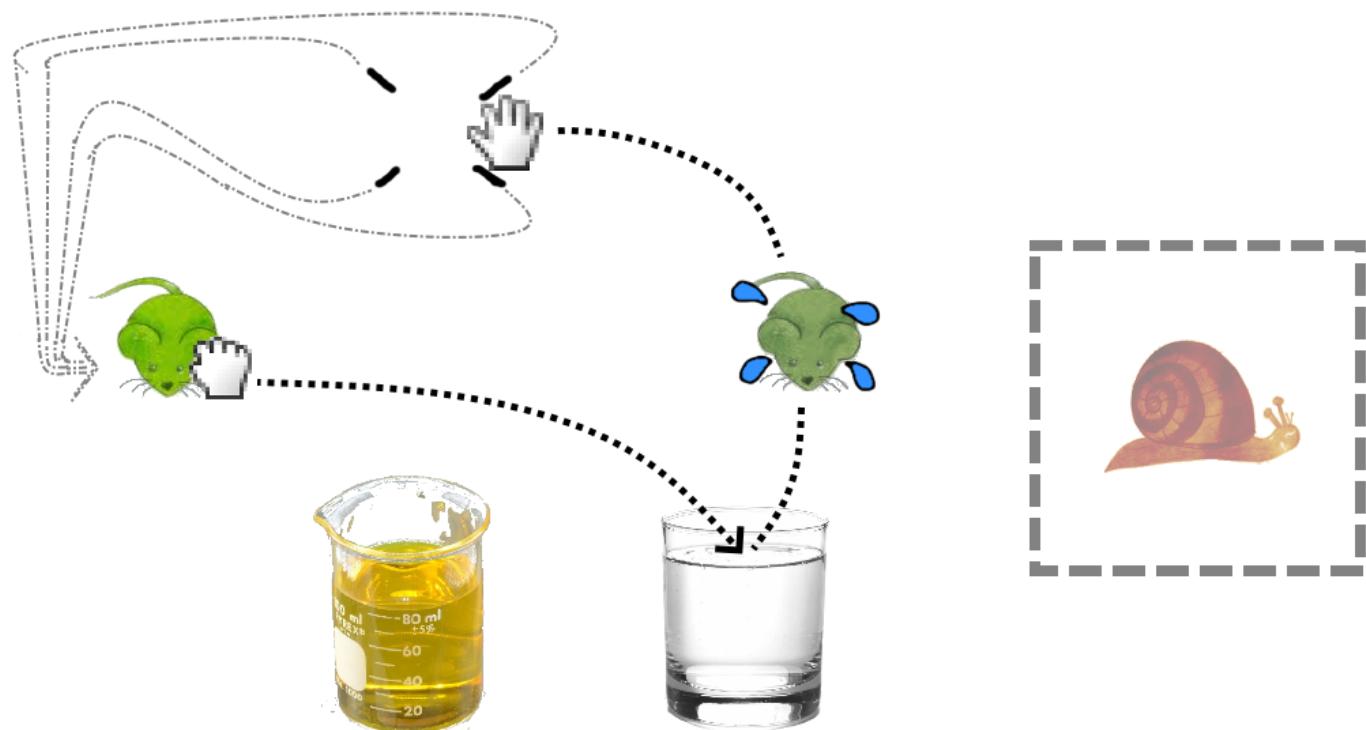


L'objet est effectivement attrapé, puis trempé dans des cuves qui le transforment. Au fur et à mesure, le concept de souris verte s'est affiné:

Malédiction de la souris verte:

Quand l'objet est lâché, il meurt.

Quand l'objet meurt, il ressuscite au point de départ.

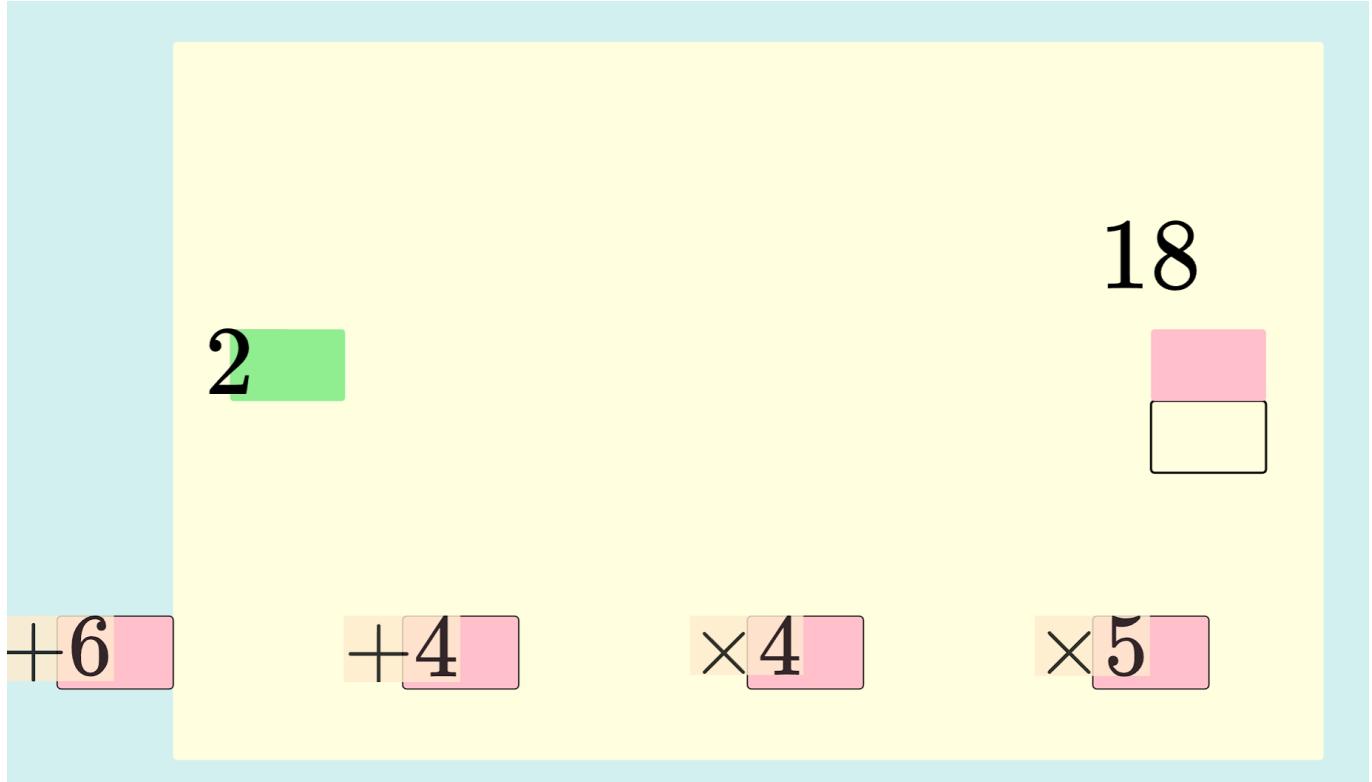


Voilà la souris verte telle qu'elle est utilisée dans le logiciel. Elle est utilisé dans un cadre bien précis: les challenges. Un challenge est un exercice basé sur les transformations: On montre à l'élève l'objet qu'il doit fabriquer, en utilisant les opérateurs de transformation qui lui sont proposées. L'objet de départ qu'il doit transformer est la souris verte. Quand l'élève a réussi le challenge, un nouveau challenge (pareil mais différent car généré aléatoirement) lui est proposé. Cet objet maudit, la souris verte, présente deux avantages majeurs en terme d'éducation: l'élève est obligé de garder la main sur la souris d'ordinateur en gardant le clic gauche appuyé, pour transformer la souris verte, ce qui réduit de 50% son potentiel de faire des bêtises, et ce qui matérialise son engagement

l'élève peut faire autant d'erreurs que nécessaire, ce n'est pas grave
s'il a appliqué une transformation qui ne lui plaît pas, il lâche la souris verte,
et elle revient à son état initial, et il peut essayer une autre séquence de transformations.

la forme canonique des challenges

voici comment est composé un challenge, la plupart du temps.



Il y a trois zones bien distinctes. L'objet initial, les opérateurs et l'objet final. Le sens de lecture en France est de gauche à droite donc le challenge va de gauche à droite mais on pourrait bien sûr l'adapter de la droite vers la gauche.

Simplifier ou ne pas simplifier

La simplification, et plus généralement la réécriture sous une autre forme, fidèle à la première, sont très importantes en mathématiques. Au niveau scolaire, bien comprendre un objet mathématique, c'est être capable de le voir sous toutes ses formes. Par exemple, on 'retravaille' une expression littérale pour mettre en avant un aspect particulier et discuter des conséquences. La simplification est une activité d'écriture qui devient petit à petit une activité mentale. Dans le canevas infini, on peut créer un opérateur correspondant à chaque réécriture, ou des opérateurs qui combinent transformation classique et réécriture (juste après). Quand on fabrique des challenges pour les élèves, on peut régler le niveau d'activité mentale attendue de l'élève, en proposant ou en ne proposant pas des opérateurs de simplification. Il semble crucial de considérer l'opération de réécriture au même niveau que les autres opérations. Aussi serait-il peut-être un jour bénéfique de voir sur des copies d'élèves: $[[5+x]] * [+ (2+x)] = [[5+x+(2+x)]]$ $[(x+2)(x-7)] * [\text{développer}] = [[x^2-7x+2x-14]]$ ou sous la forme de fonctions: $\text{ajouter2}(5) = 7$ $\text{développer}((x+2)(x-7)) = x^2-7x+2x-14$

l'aléatoire comme principe de base

La triche décérébrée est une grande entrave au bon développement des élèves. Pour les protéger d'eux-mêmes, l'aléatoire semble une bonne idée car cela les amène: soit à apprendre à se débrouiller seuls soit à apprendre à tricher intelligemment s'ils regardent sur l'écran du voisin pour comprendre comment il fait, ils

sont en pleine phase d'apprentissage (et donc c'est gagné) De plus, la facilité avec laquelle les élèves ont accès aux vidéos, sur lesquelles ils pourraient voir toutes une série d'exercices "codée en dur" résolues pour ensuite reproduire leurs solutions betêtement, rend l'utilisation de l'aléatoire incontournable, si on a envie de faire travailler le maximum d'élèves. La création d'un langage de domaine spécifique (DSL) à la création de challenges semble être nécessaire (et pourquoi pas augmenté de concepts propres à l'enseignement). La maîtrise de ce langage serait alors une compétence que l'on pourrait attendre des professeurs des générations à venir, à moins qu'une interface en langage naturel, contrôlé par la voix, suffise pour une utilisation de tous les jours...

Une nouvelle forme d'activité mathématique

En affichant sur le tableau de la salle de classe, le canevas d'un élève, on pourrait lui demander de tenter un challenge, et d'expliquer à l'oral ce qu'il est en train de faire et pourquoi il utilise tel opérateur, puis tel autre, etc. comme si c'était un exercice classique.

9. l'influence d'APL: icArray

L'influence d'APL

Il faut rendre hommage à Kenneth Iverson, qui a vraiment essayé de trouver un moyen de transmettre l'algèbre différemment, bien qu'il soit malheureusement maintenant souvent casé dans les cabinets de curiosités de l'informatique. Le langage de programmation APL, créé par Iverson, est composé de beaucoup de symboles, qui sont chacun reliés à un concept particulier (2 en fait mais faisons simple). Il y a beaucoup de similarités entre Une Interface Utilisateur (AUI) et APL car les deux sont: très visuels (c'est-à-dire pratiquement sans texte) basés sur des chaînes de transformations parti d'une volonté de faire de l'algèbre différemment. Tous les concepts présentés jusque là ont comme finalité la célébration de la créativité et de l'ouverture à de nouveaux concepts. Une des façons digestes de présenter un nouveau concept (mathématique) est présenter une de ses transformations caractéristiques sous forme d'un opérateur unaire, qui va transformer l'objet en entrée en un autre objet en sortie. Mais ensuite, pour pouvoir comprendre dans des cas de figure moins dirigés, en expérimentant plus librement, il faut souvent partir de 2 objets en entrée (ou plus), qui vont se transformer en 1 objet (ou plus) en sortie. On peut utiliser des opérateurs à arguments dormants (sleeping args) pour y parvenir, mais on peut aussi utiliser les tableaux.

Les tableaux:

L'astuce mathématique pour continuer à faire le raisonnement "j'ai un objet qui se transforme en un objet" est de considérer les objets en entrée dans leur totalité. Au lieu de dire "2 brins de folie, traversant le chien et la chienne, ont donné 3 chiots", on dira "la pelote a donné une portée". Autrement dit, un tableau en entrée s'est transformé un tableau en sortie. Le canevas infini présente un avantage net sur les autres interfaces graphiques : on peut manipuler des objets de taille quelconque. Si l'objet apparaît trop grand à l'écran, on dézoomé avant de le saisir, on zscrollle pour le ramener au niveau de zoom souhaité, en version petite. Dans Une Interface Utilisateur (AUI), il y a des composites "tableau". Pour garder une correspondance directe avec le code informatique textuel, les tableaux sont arrangés de manière horizontale, le premier élément à l'extrême gauche, le dernier à l'extrême droite. On peut donc saisir et transformer des tableaux à l'écran et il y a des transformations qui tirent parti du côté visuel de l'interface.

Fabriquer des tableaux: à la APL:

La fonction iota de APL est très intéressante pour habituer les élèves aux tableaux. $[[5]] * [i] = [[1 2 3 4 5]]$ une façon naturelle: Les tableaux sont très utiles à manipuler, il faut donc pouvoir les fabriquer facilement. Encore une fois, Une Interface Utilisateur (AUI) tente de s'aligner sur les manipulations basiques de la vie. Pour avoir trois noisettes dans la main, les enfants en prennent une avec pouce et index, la basculent et la gardent saisie dans leur paume avec les autres doigts, et font pareil pour la seconde et la troisième. Ce qui représente la saisie continue d'un objet est le maintien appuyé du clic gauche. Heureusement, la souris a 2 clics. Le mouvement va donc être le suivant, on saisit le premier objet avec le clic gauche et on reste appuyé. On ajoute le deuxième objet au tableau en faisant un clic droit quand on se trouve au dessus : un tableau à 2 éléments est maintenant saisi. On continue ainsi de suite. On pourra alors utiliser un opérateur qui considère le tableau comme un seul objet ($[[2 3 4]] * [+] = [[9]]$), ou un opérateur qui travaille seulement sur l'élément du tableau qui l'a touché (voir l'exemple).

10. Quand le prof manipule: eduDemos/warp/donutOperators

eduDemos:

Hors de phases d'exploration autonome, l'élève peut bien sûr compter sur son professeur pour lui transmettre les concepts. Une Interface Utilisateur (AUI) est un conteneur d'interactivité. Elle peut servir au professeur, comme un super tableau blanc, à créer, en direct, devant les élèves, des environnements dédiés à tel ou tel concept de tel ou tel chapitre du cours. Ces situations sont appelées eduDemos. Elles combinent des manipulations du professeur et des parties automatisées. Elles ont chacune leur propre disposition graphique. Le professeur peut écrire, mettre des flèches, pendant la présentation. Le professeur peut également réagir aux questions des élèves en essayant de faire ce qu'ils proposent et voir ce qu'il se passe (une exploration avec un guide). Par exemple, changer des valeurs, ajouter d'autres objets, ... La conception d'eduDemos de qualité semble être une bonne ouverture, une bouffée d'air pour les professeurs qui ont envie de faire les choses à leur manière.

Dans les logiciels de géométrie dynamique, les objets sont les points, les segments, les cercles, les courbes... Ces logiciels sont très puissants pour faire surgir des motifs dans nos cerveaux : en faisant varier certains paramètres, on voit ce qui change et ce qui ne change pas (différent mais finalement pareil). On comprend alors ce qu'est la médiatrice d'un segment, en faisant varier le segment. Pourquoi ne pas généraliser ces outils pour les nombres et les structures informatiques de base ? Pourquoi ne pas généraliser ces outils pour les mots et les textes ? Le parti pris est le suivant: en ayant un seul et même outil, un seul et même cadre (bien qu'infini) pour manipuler nos outils conceptuels de base, on ne fragmente pas la connaissance, on la consolide. Tout cela découle du canevas infini: puisqu'il peut contenir une quantité d'information illimité et donne l'utilisateur un moyen facile d'y naviguer, plus besoin de cloisonner.

Un exemple de concepts émanant de la préparation d'une eduDemo: la commande !warp! et les mixedScale objects

Il n'y a plus de raison de se priver d'un espace généralisé pour les mathématiques, où l'on pourrait étudier des fonctions, tracer des figures géométriques, faire des calculs, dessiner des graphes, écrire des algorithmes et les exécuter, manipuler des propositions logiques, créer de nouveaux symboles, et d'autres objets encore inconnus. La création d'une eduDemo passe souvent par la décomposition d'une activité élève en sous-parties. Le fait d'associer toute transformation à un opérateur amène à expliciter le processus de transformation de l'information, de l'énoncé jusqu'à la solution complète d'un exercice par exemple. Voici un exemple concret: Quand les élèves apprennent la numération, on leur présente la demi-droite réelle. On leur demande de repérer des points sur cette droite à partir de leur abscisse. Si je te donne le nombre 4, tu le

places où sur cette règle graduée ? Si je ne laisse que les graduations pour le 0 et le 1, est-ce que tu arrives encore ? La notion de la nature d'un objet est importante dans l'apprentissage des mathématiques. On passe du temps pour expliquer aux élèves qu'un point, ce n'est pas un nombre. Il est cependant, à mon sens, dommageable de contraindre, a priori, la transmission par une complexité qui n'a, encore une fois a priori, pas lieu d'être. L'opérateur [warp], ou la commande !warp! (Touche E), permet d'envoyer un nombre sur la droite réelle, centré sur le point dont il est l'abscisse. On imagine facilement une activité où le professeur préparerait une liste de nombre, et après une concertation avec ses élèves, utiliserait la commande !warp! pour voir où le nombre se serait téléporté. Certains nombres, comme 100, seraient dans un premier temps introuvables, mais seraient retrouvés après plusieurs déplacements panoramiques ou un dézoom/zoom. Puis cela amènerait la discussion sur la position précise des nombres et la nature particulière du point.

11. Galaxies d'apprentissage (plus on zoome, plus on se spécialise) (dans iC et dans 3dRoom), cartographie de scolarité, liens entre disciplines

Vue de loin, une carte qui ressemblerait à un arbre, une gigantesque carte mentale, pourrait accueillir, en complément des cours en classe, une collection canonique des savoirs qu'un élève rencontre dans sa scolarité. Il y aurait une branche pour les mathématiques, une branche pour le français, etc. Dans les maths, une sous-branche pour l'algèbre, une sous-branche pour la géométrie, etc. Puis des portails assureraient la destruction de l'ordre apparent ("topologisation") en mélangeant les sous-branches, les disciplines, etc.

Scolarité en un clin d'oeil:

L'élève pourra rapidement visualiser les zones qu'il maîtrise bien et les zones d'ombre. La progression scolaire de l'élève pourra toujours être améliorée. Le professeur pourrait mieux diriger son enseignement, d'après les cartographies de ses élèves (qui sont comme des infographies dynamiques). Mais ce côté cybernétique n'est pas la raison d'être d'une Interface Utilisateur (AUI). On pourrait s'en passer. (La raison d'être du logiciel est de permettre la manipulation de tous les objets mathématiques, aussi complexes puissent-ils être, afin que le tryptique "Manipuler, Verbaliser, Abstraire" soit toujours réalisable. Ce qui revient aussi à remettre du Manipuler après l'Abstraire (boucle) : Manipuler, Verbaliser, Abstraire, Manipuler, Verbaliser, Abstraire, ...).

Mémorisation:

Les élèves pourront mémoriser plus facilement car ils spatialiseront les chapitres et les exercices. Ils pourront aussi faire des dessins pour égayer et s'approprier leur univers d'apprentissage, d'expérimentation et de création. Refaire le chemin, de la carte affichée complètement à leur zone de travail courante, en passant par les zones intermédiaires, réactivera les concepts presque subliminalement.

cours à distance, "localisés":

Un professeur pourra se connecter à la galaxie de l'élève et le guider, lui expliquer différemment. Bien que l'interaction soit à distance, l'élève et le professeur seront dans un espace familier sur lequel l'un et l'autre auront la possibilité d'agir.

chasses aux trésors (TH (treasure hunt)):

Comme pour une chasse au trésor dans la vraie vie, on commence par lire un indice, puis, en réfléchissant et en tatonnant, on se déplace et on arrive au deuxième indice, etc. Dans le canevas infini, la zone de recherche est beaucoup plus grande, puisque le déplacement est beaucoup plus efficace. On sait que l'on a trouvé

quelque chose d'intéressant quand on zoome et que l'on voit quelque chose grossir à l'écran. Quelques conséquences: c'est très engageant car cela fait appel à nos sens et qu'il n'y a pas de danger (à par se perdre dans l'infini) pas besoin d'ajouter d'objets en mouvement: Un environnement totalement statique peut héberger une chasse au trésor. Ce n'est pas l'environnement qui change, c'est l'état cérébral du joueur qui se déplace dedans (il a de plus en plus d'informations). on peut aussi utiliser la galaxie d'apprentissage comme zone de chasse au trésor et amener les élèves à traverser des zones de cours, et rafraîchir leur mémoires au passage. On peut disposer les éléments de la chasse au trésor de manière à ce qu'ils soient, à première vue, invisibles, et ainsi éviter de polluer des zones qui doivent rester assez propres. il faut vraiment chercher car un scan exhaustif en zoomant/dézoomant sur chaque portion du monde serait bien trop long et trop incertain. Comme pour les challenges, les professeurs (ou les créateurs de jeux) peuvent laisser s'exprimer leur propre style et leur créativité pour créer des chasses au trésor très engageantes pour les élèves.

12. Divers:

autre modes d'interaction: laisser pour un temps le clavier et la souris

Entendre et écouter:

En utilisant la webcam des ordinateurs, on peut interagir différemment avec le canevas infini. Un objet icWebcam peut afficher ce que filme la webcam. Un objet icEye peut essayer comprendre ce qui est affiché dans une zone du canevas. C'est un peu comme la différence entre entendre et écouter attentivement, ici ce serait plutôt voir et détecter. On peut fabriquer autant d'icEye que de façons de détecter.

get100 (et autres conséquences des QRCode)

Si l'icEye est réglé pour détecter des QRCode et qu'à chaque QRCode, on associe une commande, alors on peut passer des QRCode devant la webcam de l'ordinateur et voir des conséquences directes dans le canevas. De plus, si l'on affiche le canevas sur un smartphone, en utilisant des cartes à jouer dont le verso est un QRCode, et le recto est la façade d'un opérateur (par exemple [+ 2]), on peut créer des jeux de cartes basés sur la transformation d'objets dans le canevas. Get100 est un jeu de cartes à 2 joueurs où chaque joueur essaie d'atteindre le premier 100, en scannant les cartes d'opérations qu'il a en main, sur le smartphone qui héberge le jeu.

alicelInjection: traverser l'écran, passage réel -> virtuel

Pour faciliter la création d'objets dans le canevas, par exemple par les enfants, on peut imaginer de connecter (de façon simple, par QRCode par exemple) ordinateur et smartphone (via une connection bluetooth par exemple), pour tirer partie de la caméra du téléphone. En filmant le canevas affiché sur l'écran d'ordinateur, tout en étant connecté à celui-ci, on peut facilement imaginer de faire traverser l'écran d'ordi à des objets posés sur l'écran. Exemple: On prend une carte à jouer "5 de pic" réelle et on la pose sur l'écran filmé par le téléphone: Un objet "5 de pic", formé avec l'image filmée, sera maintenant dans le canevas. On appelle cela une alicelInjection (on est passé de l'autre côté du miroir). Un enfant pourrait commencer un dessin sur du papier réel avec des feutres réels et, après lui avoir fait traverser l'écran, continuer avec les outils du canevas infini. Technologiquement parlant, on pourrait intégrer des "caméras miroirs" aux ordinateurs, pour éviter d'avoir à utiliser le smartphone. C'est parce qu'une Interface Utilisateur se veut sans menu, qu'il faut trouver des solutions innovantes d'injection de contenu.

théâtre/marionnettes virtuelles: Une pièce de théâtre ou un dessin animé en direct

On suppose que des acteurs sont connectés et qu'ils évoluent dans un même canevas infini hôte. On suppose que les spectateurs ont un point de vue limité : leur screenPov est fixe et ils ne peuvent pas voir "les coulisses". Chacun des acteurs contrôle une marionnette, avec le clavier et la souris, et cela en utilisant l'interface utilisateur habituelle (on saisit avec la souris, on utilise le colorScroll, on touche des opérateurs, ...) Les marionnettes entrent sur scène, quittent la scène (c'est-à-dire la zone visible par les spectateurs). Grâce aux opérateurs, des parties scriptées (totalement automatisées) peuvent se dérouler. En coulisse, des assistants peuvent aider les acteurs à changer les costumes et les accessoires des marionnettes, à modifier la scène et animer des objets, et cela, encore une fois, en utilisant l'interface utilisateur habituelle. Une fois maîtrisés les bases de l'interface utilisateur, les enfants peuvent s'essayer à devenir marionnettistes, utiliser leurs propres dessins comme décors/personnages, et bien d'autres fantaisies issues de leur imagination.

III. Remarques

Un point sur le code:

Le logiciel a été codé en javascript, parce que c'est le langage de M. Toutlemonde. A ce titre, le logiciel pourrait bénéficier du portage en javascript de certaines bibliothèques ("libraries") de code (par exemple, de régression symbolique). Le but visé n'est pas la performance mais la manipulation et l'appropriation. L'étude (même très superficielle) des bibliothèques de code servirait à faire la transition vers la programmation textuelle.

Remarque bizarre autour de la densité d'information:

On peut créer une version du canevas infini avec un niveau de zoom maximal, qui reviendrait à réaliser l'hypothèse d'une densité maximale de l'information. Les objets seraient cantonnés à un intervalle de niveau de zoom correspondant à l'information qu'ils contiennent : Il y aurait un niveau de zoom plancher pour chaque objet. Les objets riches en informations vivraient dans les hautes strates. Mais, bien sûr, mises à part certaines transformations, au niveaux de zooms "habituels" d'utilisation, cela ne ferait aucune différence pour l'utilisateur.

Conclusion/Lyrisme:

J'espère que vous avez compris l'apport majeur du canevas infini et des fonctionnalités qui en découlent naturellement. J'espère que vous avez expérimenté mentalement les transformations, les déplacements qui s'effectuent normalement sur un ordinateur, avec un clavier et une souris.

Si vous voulez voir si une de vos idées géniales, qui vous serait apparue en lisant le livre, a déjà été capturée et couchée sur le papier, ou si, tout simplement, vous êtes curieux de voir les autres concepts, écrits dans un style plus brut, il y a still_shute.txt (le mieux c'est de commencer à partir de la ligne 3090, "20 février 2021", avant c'est un peu trop magmatique). Il y a beaucoup beaucoup de concepts, et ce livre ne contient que ceux qui semblaient les plus intéressants (tant pis pour ceux qui sont passés à la trappe).

Ce livre se veut être une invitation. Il reste encore énormément de travail. Il reste encore énormément de concepts à découvrir. Tant mieux. Si on veut que les élèves ou finalement tous les ordinateuristes en bénéficient au plus vite, cela passera nécessairement par le travail d'une équipe (et de doctorants...), et on pourra alors voir quels seront les impacts d'un tel logiciel, voire d'un tel OS.

Le logiciel commencé (3dRoom+ic.html) peut servir d'inspiration pour la suite. Un portage, entièrement en 3D, est tout à fait envisageable (ce qui demandera de repenser ou de généraliser certains concepts). Juste à titre d'exemple, la programmation par le dessin (2D) deviendrait "la programmation par les tuyaux" (3D) où un ensemble de programmes s'apparenterait à une raffinerie (on reste dans la chimie).

Pour ceux qui veulent simplement utiliser le logiciel: à vous de créer vos dessins, vos présentations, vos cours, vos exercices...