What is a beam line in the silicon brain of the computer?

Etienne Forest

(フォレ　エティエンヌ)

KEK

What is a beam line in the silicon brain of the computer?

Etienne Forest, KEK, Japan

Main points and assumptions of my talk

1) *Nearly all computer codes* in accelerator physics use a technological prejudice: the lens paradigm. We track from one surface to another. *Even space charge* codes use this approach although it is *incorrect* strictly speaking in the presence of collective effects.

2) This prejudice is grounded in a very reasonable assumption: most of our machines are made of magnets whose *properties are designed to be indepedent of the system in which they are embedded*--- like camera lenses. This is a wish, not a physical necessity. But it is almost true in our machines.

3) So, I accept the prejudices like all other lemmings.

4) With modern computer languages, we ought to project this lens approach on the silicon brain with maximal efficiency. For that purpose I joined the "C++" project that became Classic and later MAD9. I left it immediately when the SLAC people claimed that they knew what a "beam line" was. They did NOT. They simply decided to implement the old ideas in C++. Fine, no attempt at improving upon our misconceptions, so I lost interest.

5) In 2000, I decided to revisit this issue because I learned FORTRAN90, a language with pointers (and operator overloading). I was at KEK doing nothing useful, so why not?

6) So, I wrote a code, which I called "Small Code", which eventually had structures capable of handling almost arbitrary complex beam lines within the ubiquitous lens paradigm.

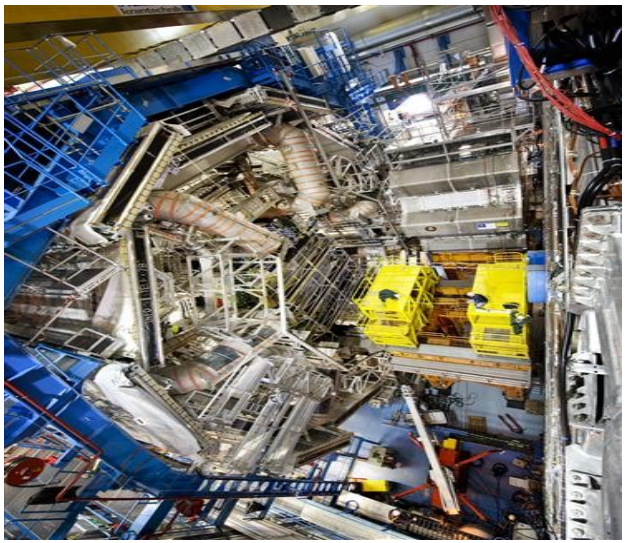
7) I give you here my *personal solution* to this problem. *There are many possibilities*. But the standard way of viewing a beam line as a sequence of magnets is certainly an incomplete and ultimately incompetent projection of our mathematical structures on the silicon brain (given modern languages like C++). This should have been understood in 1992. But, MAD9 and MAD-X later, too late....

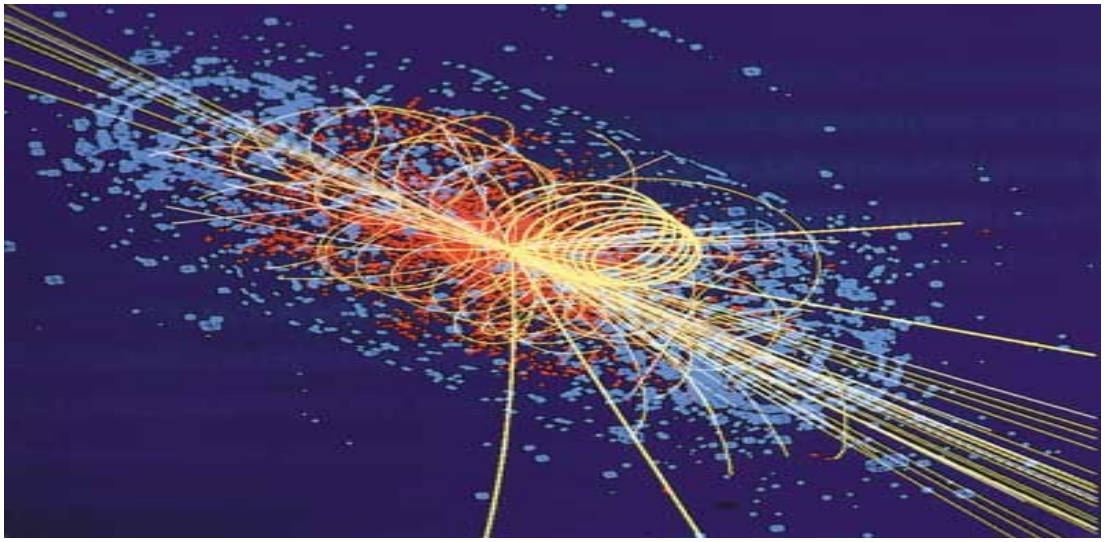
The lens paradigm

Technological prejudicial assumptions shape particle tracking in Accelerators

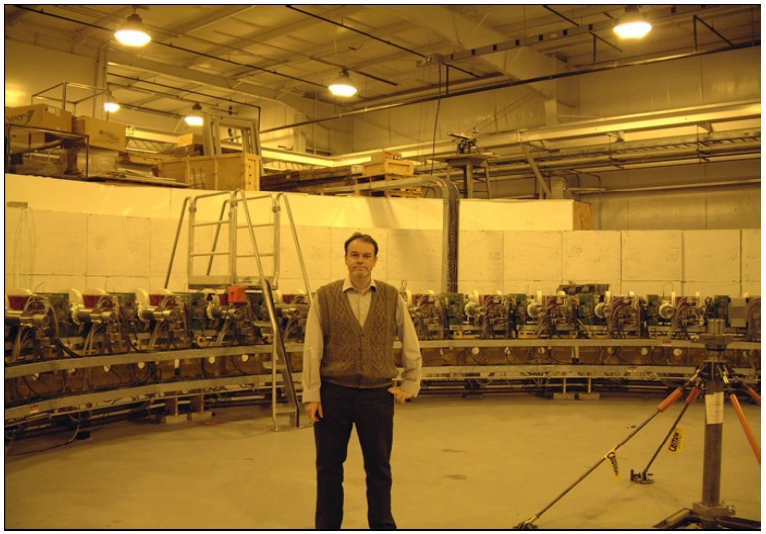
Constrasting Physical Objects

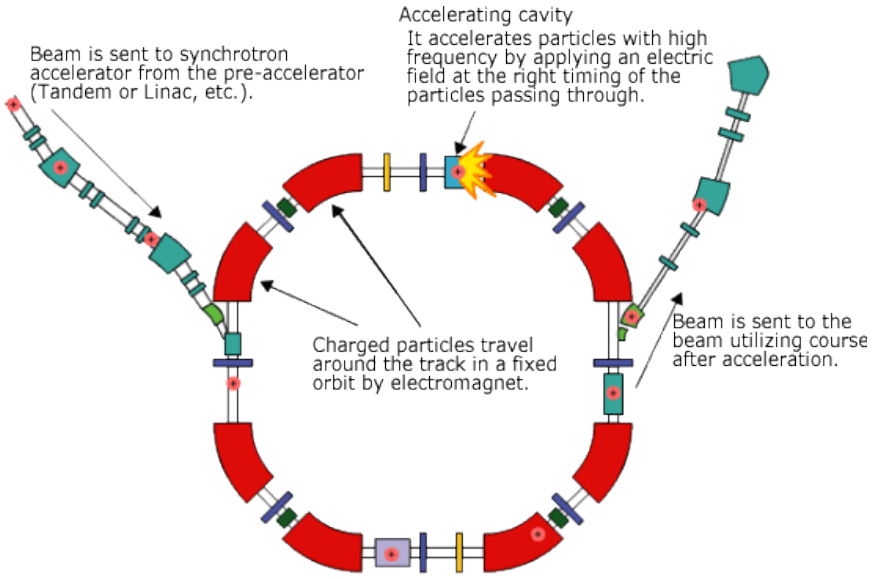
- Ordinary devices (no prejudices)





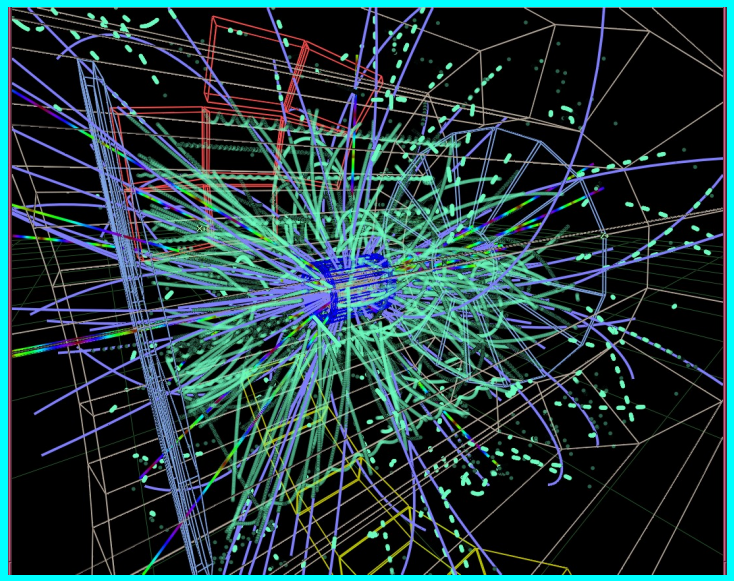
- Devices thought to be made out of lenses



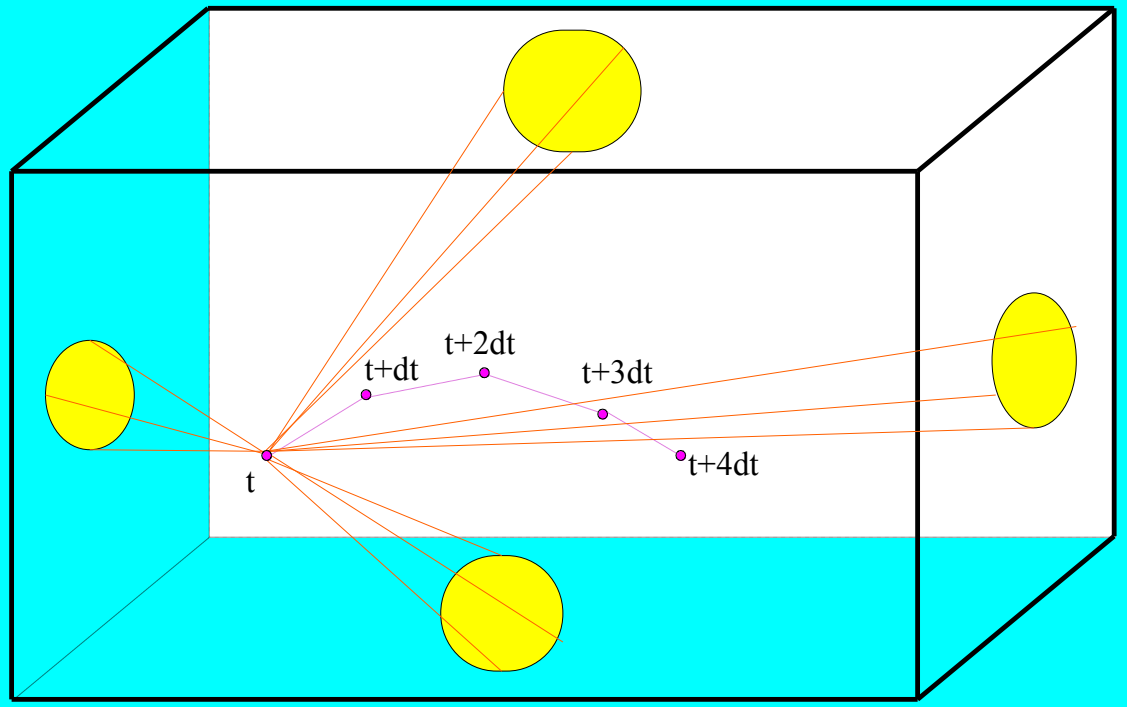


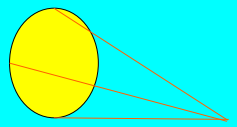
Example of ordinary device: Detector

1. The field at one point is influenced by all parts of the device
2.All particles trajectories are potentially interesting: from left to right, right to left, top to bottom, etc...
3.Various particles are produced; you cannot specialize to a given mass, a given charge or a given energy.



Real simulation





The yellow areas schematically represent coils or metal which produces the B-field in the device.The orange lines represent symbolically the "action at a distance."

Therefore, the proper strategy is to apply the laws of physics ***free of any technological prejudices***, as they are taught in a true scientific context. Let us call this the ***"standard strategy."***

Summary from a programming standpoint

The Electro-Magnetic field is the Central Object of the Theory!

It should be the central class of your software!

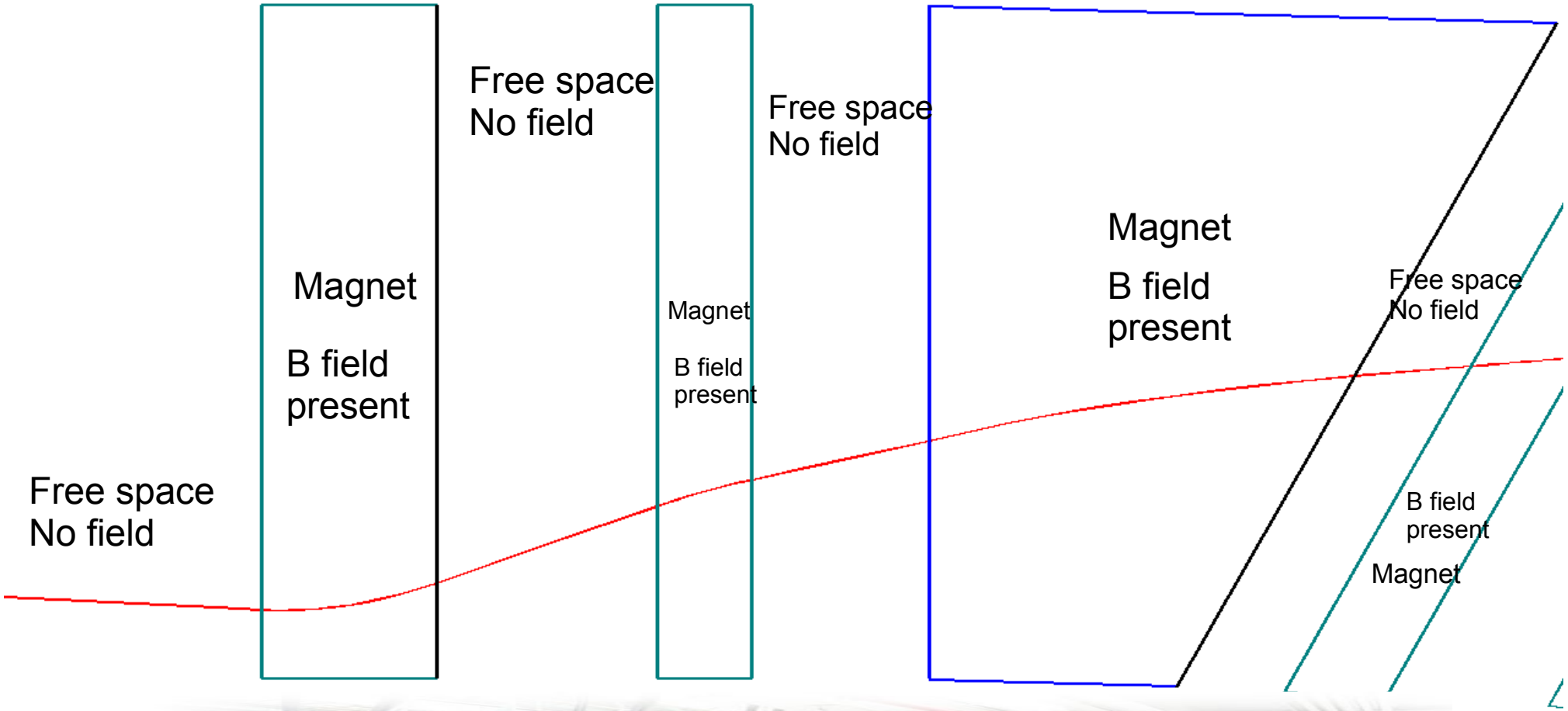
$$\frac{d\vec{x}}{dt} = \vec{v}$$

$$\frac{d}{dt} m\gamma\vec{v} = q\,\vec{v} \times \vec{B}$$

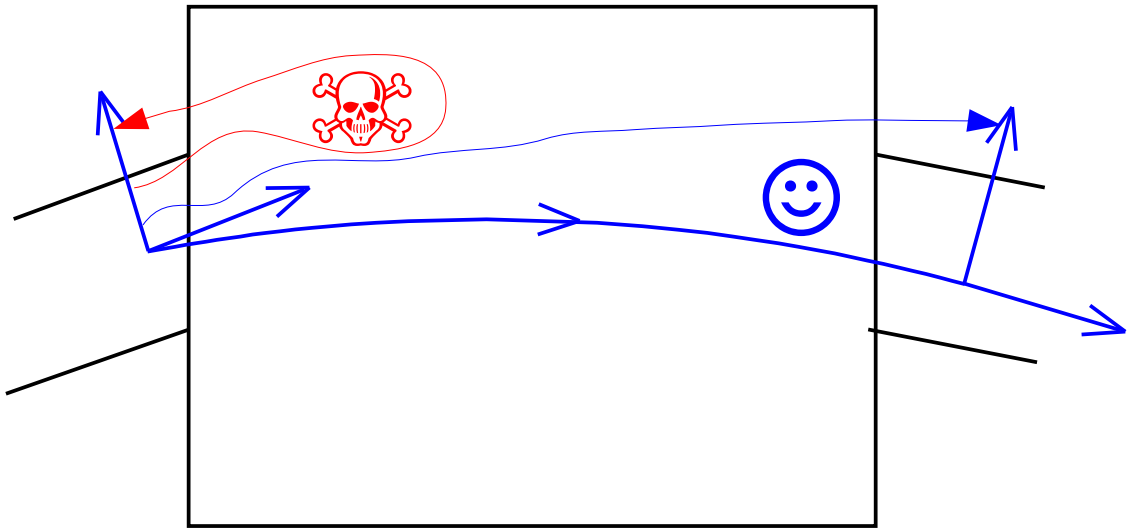
Not so with accelerators physics!

Why is that so???Next slides!

Lens type device

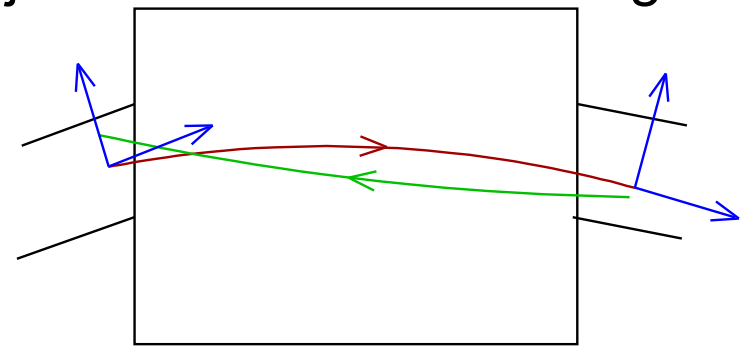


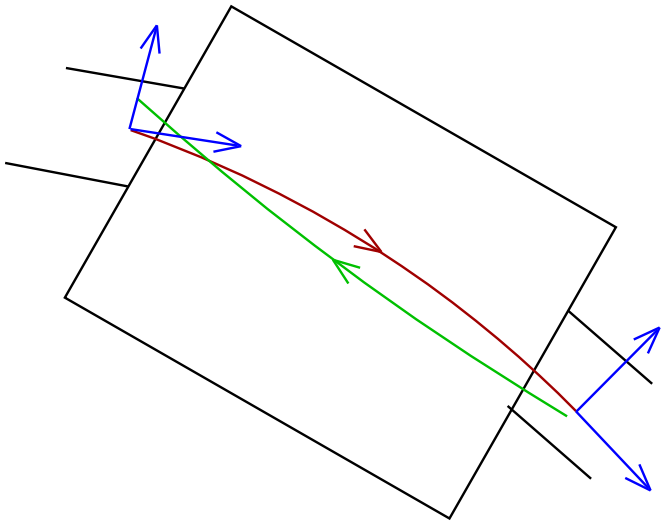
Lens Paradigm



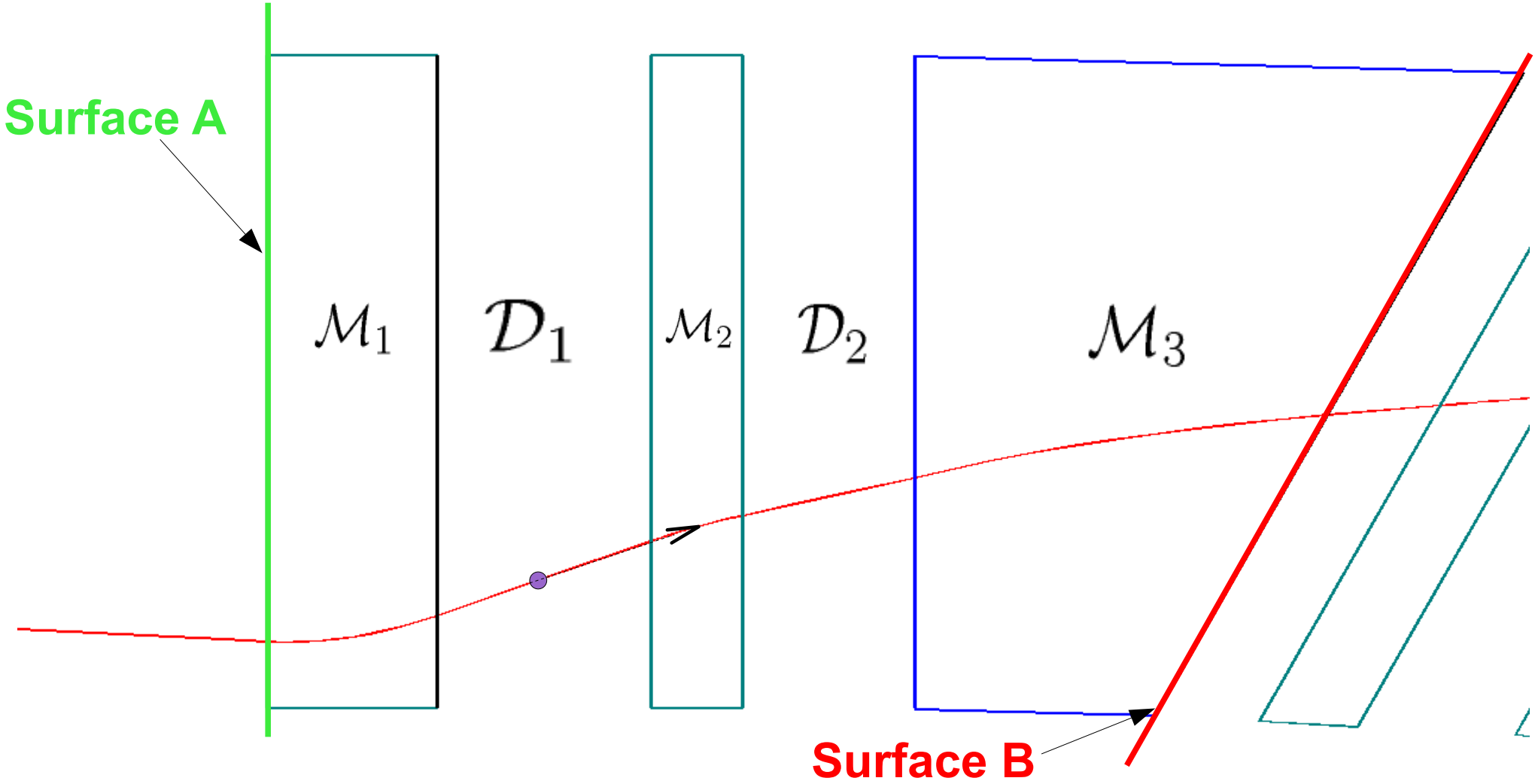
Not all trajectories considered

Trajectories attached to magnets





First Trivial Consequence



Accelerator physicists write the "map" from surface A to surface B as the concatenation of the maps of the individual magnets and the empty space drifts.

$$\mathcal{M}_{AB} = \mathcal{M}_3 \circ \mathcal{D}_2 \circ \mathcal{M}_2 \circ \mathcal{D}_1 \circ \mathcal{M}_1$$

Magnet based propagators

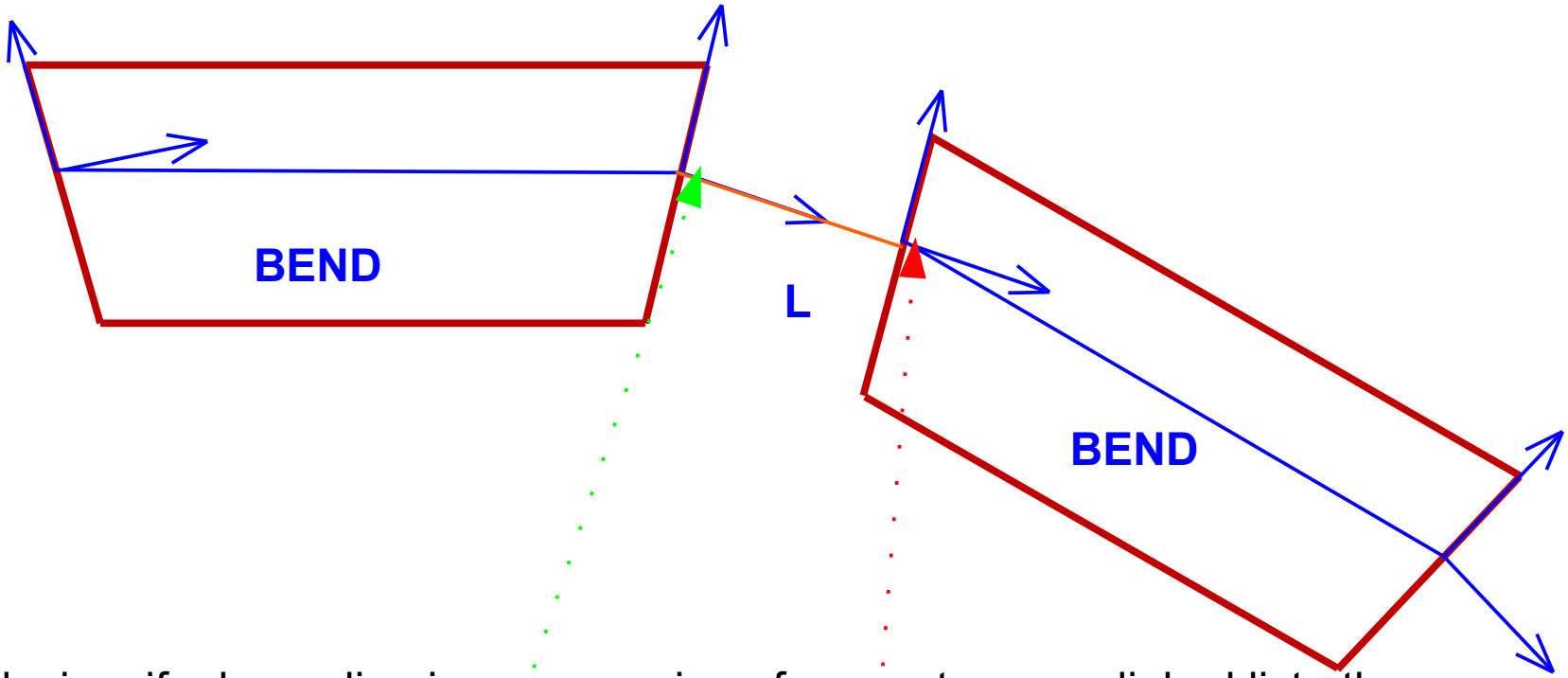
Magnet (viewed as hardware) $\rightarrow$ propagator $\rightarrow$ Dynamics

Local B field produced by magnet

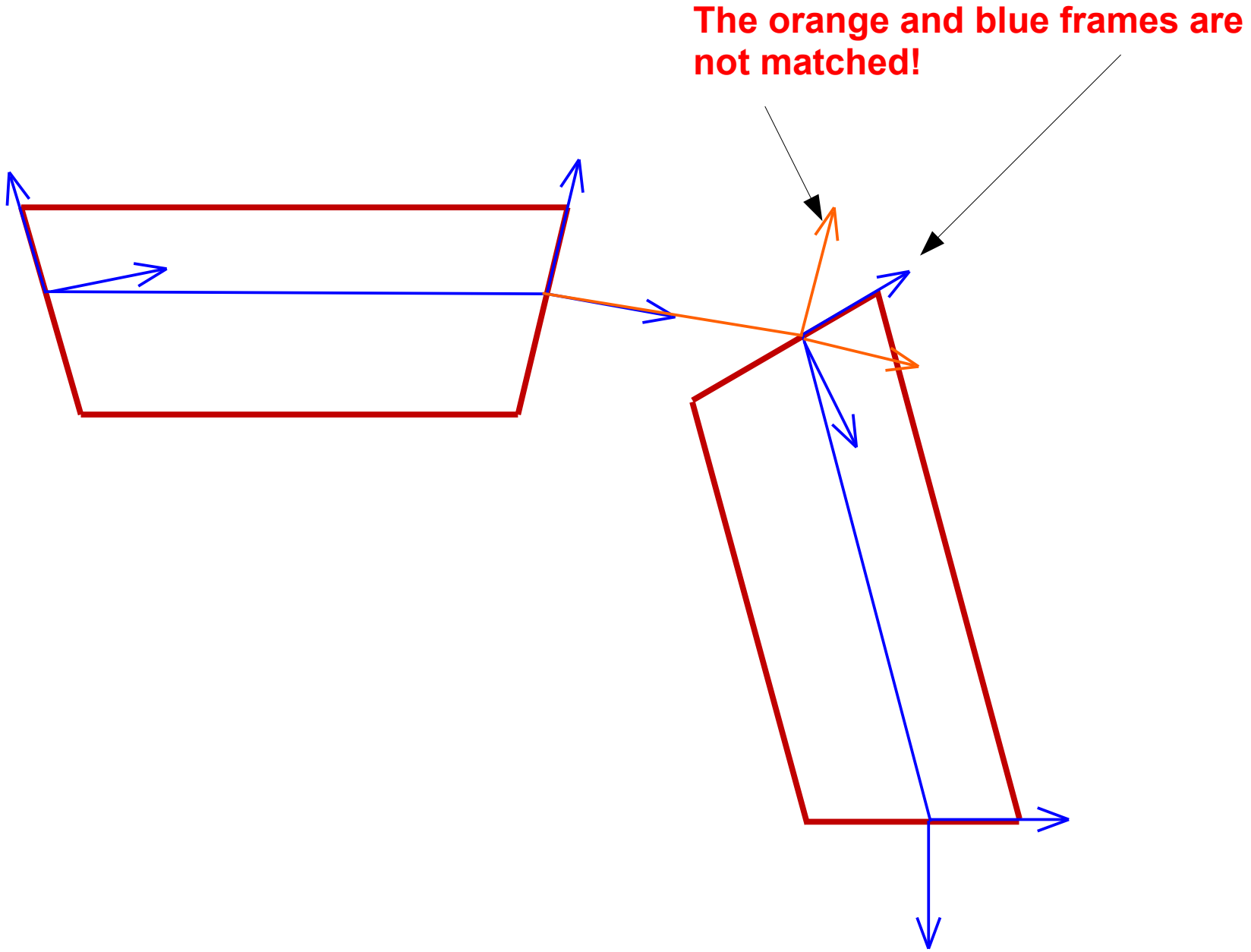
Ergo, the most unsophisticated view of accelerator physics applies this view trivially, i.e., a beam line is an ordered sequences of magnets and their propagators.

*A huge C++ project, called CLASSIC, which eventually collapsed, **did not go** beyond this. I hope to succeed in explaining how one needs to go beyond this to exploit the lens paradigm to its fullest.*

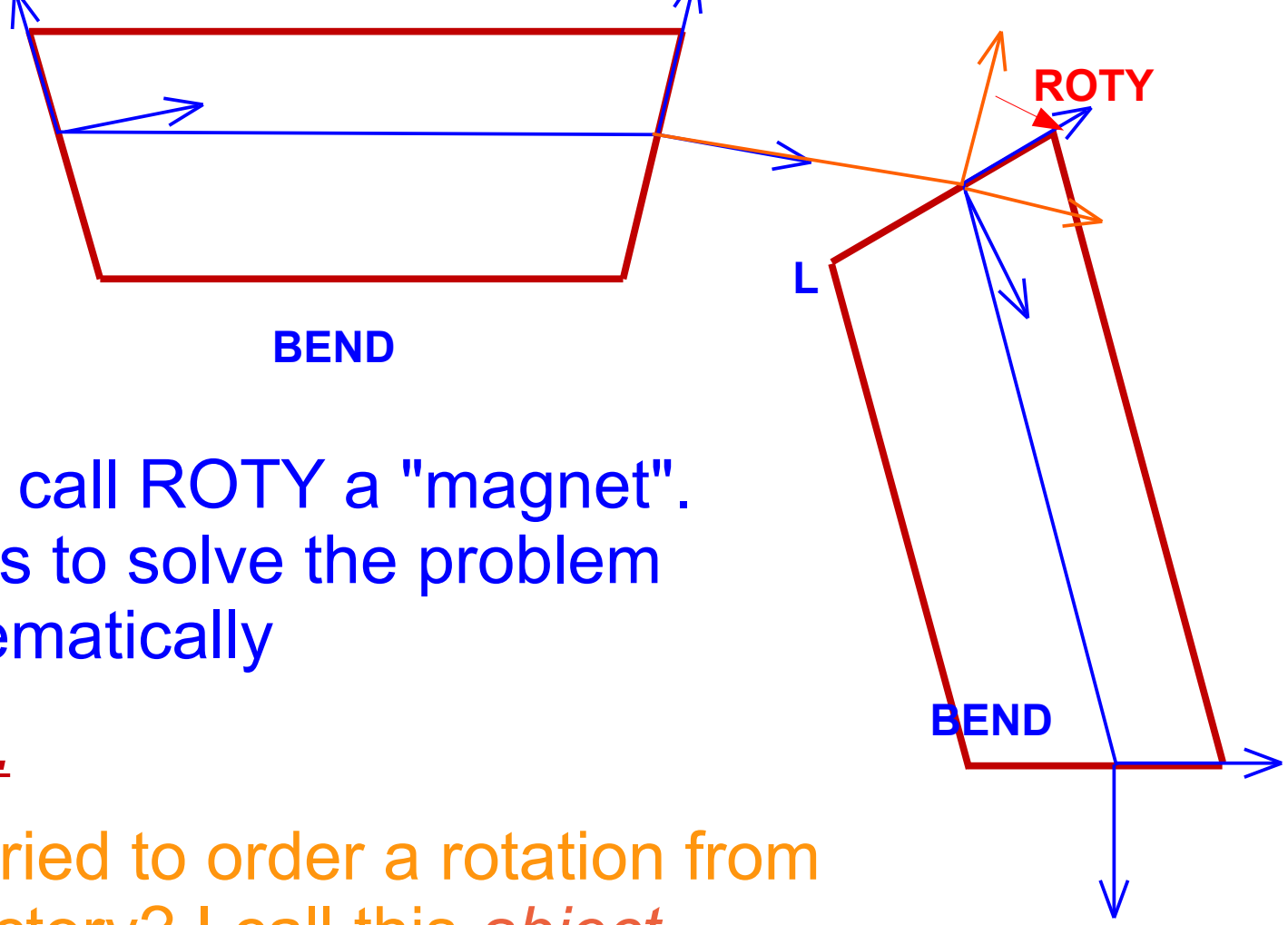
```
! INPUT FOR PROGRAM MAD-X OF CERN


   L : drift, L= 0.2;
   BEND : RBEND,L= 2.0, ANGLE=2.0*pi/10.0;


beamline : line = (BEND,L,BEND);
```



Conclusion: if a beam line is a succession of magnets, say a linked list, then the frame of reference at the end and at the start of magnets must be aligned!

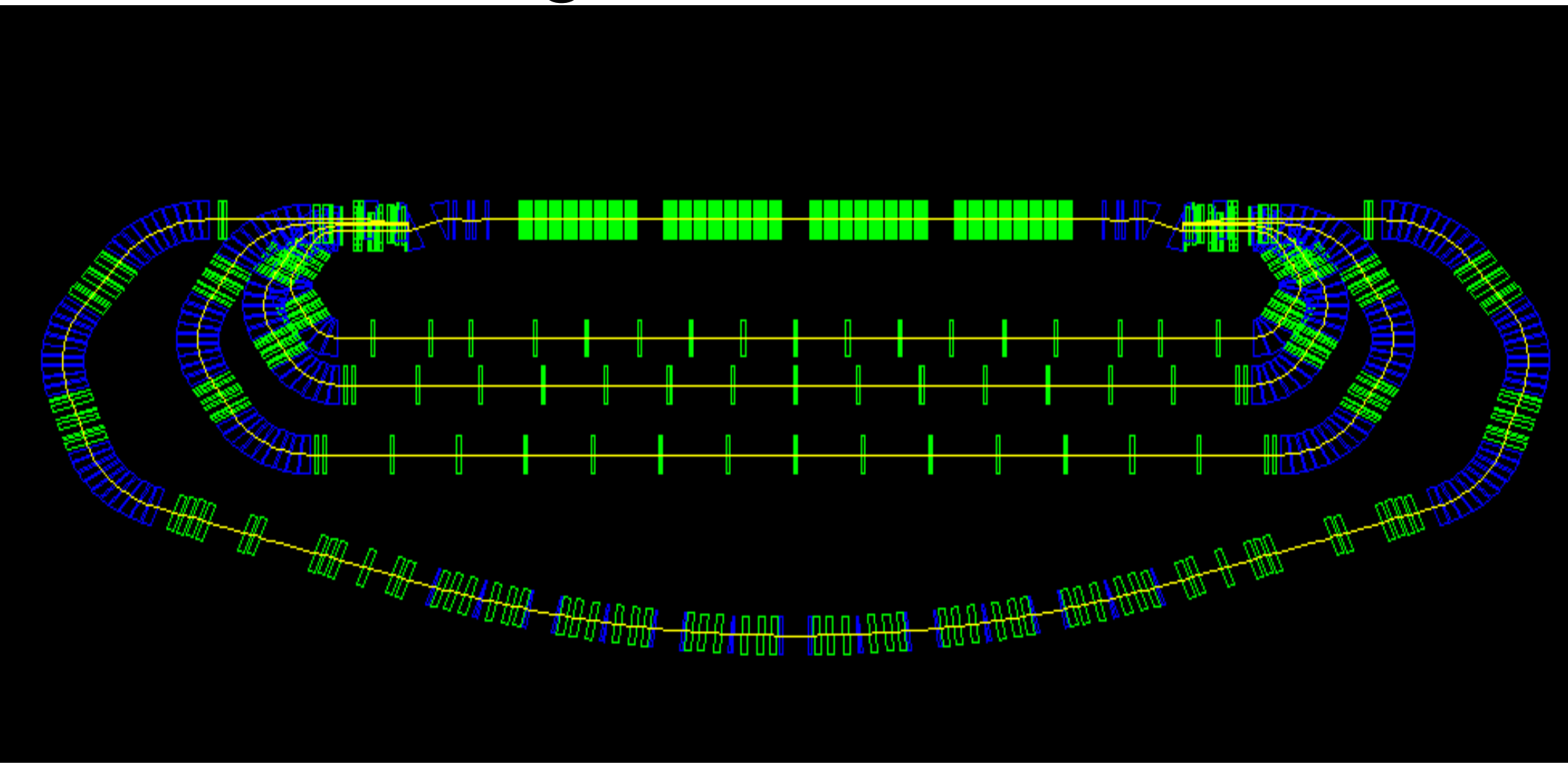


We should be able to simulate this situation!



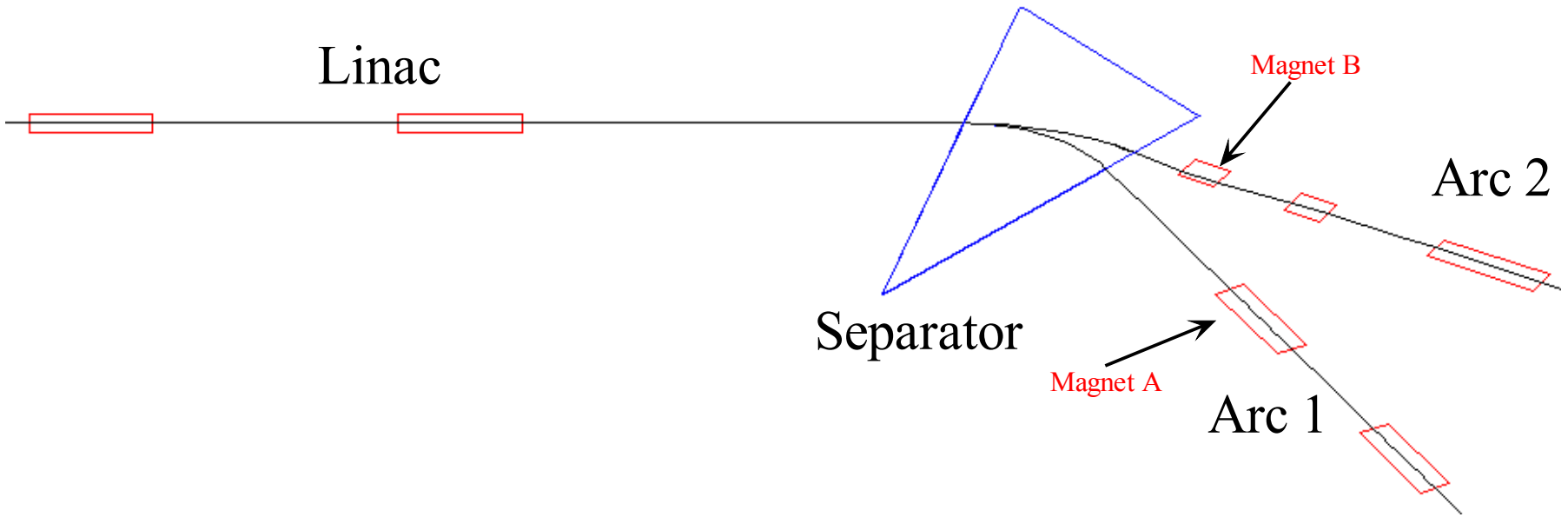
- Could call ROTY a "magnet". Seems to solve the problem mathematically
- ***But....***
- Ever tried to order a rotation from the factory? I call this *abject* oriented programming!☹💣
- There is a more fundamental problem this approach *does not* solve : next topic!

Examining Possible Structures

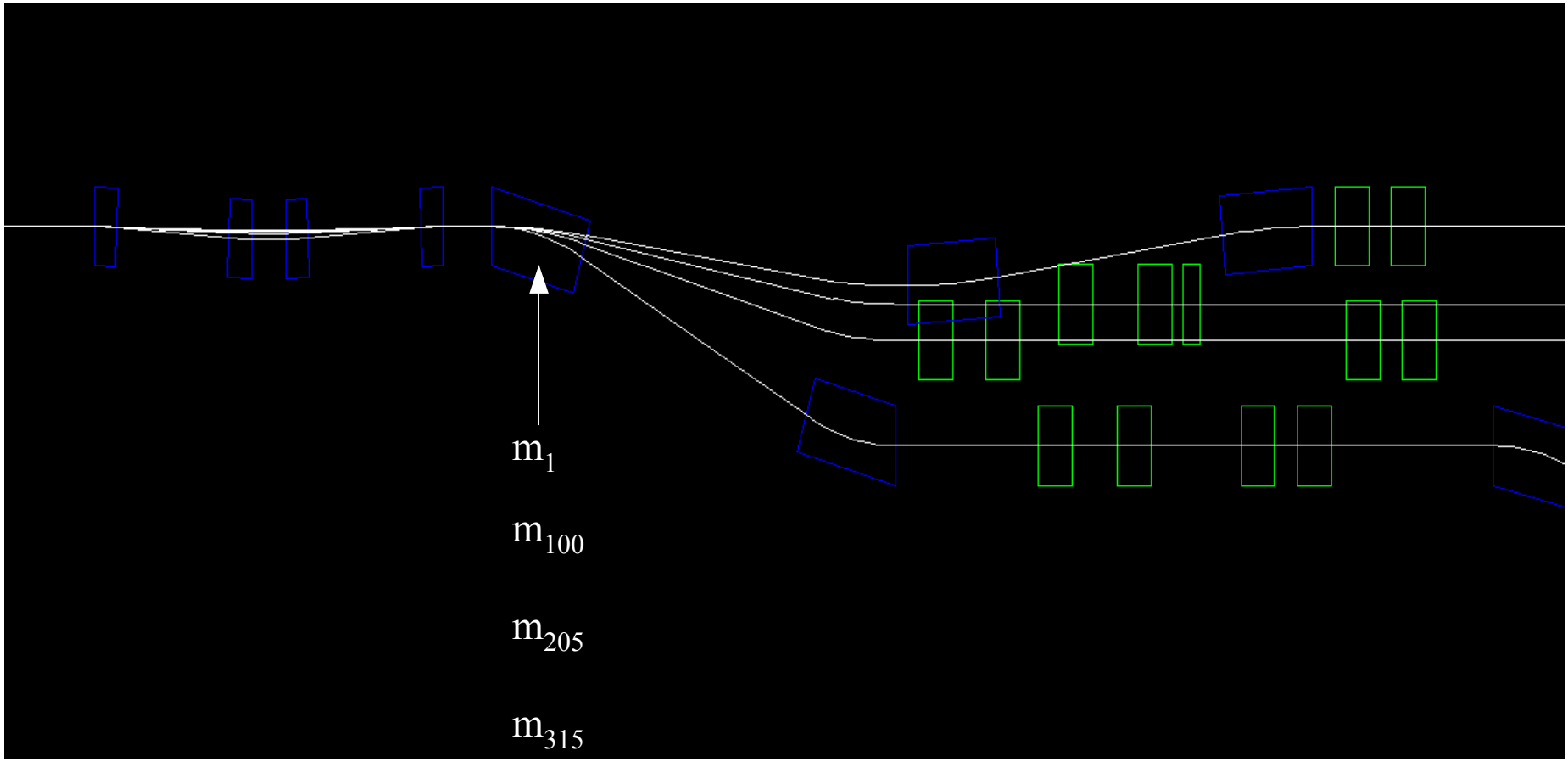


$R=(m_1,m_2,m_3,\ldots\ldots,\ m_n)$?

Detail View of Problematic Region



Sequence Magnets (Propagators)



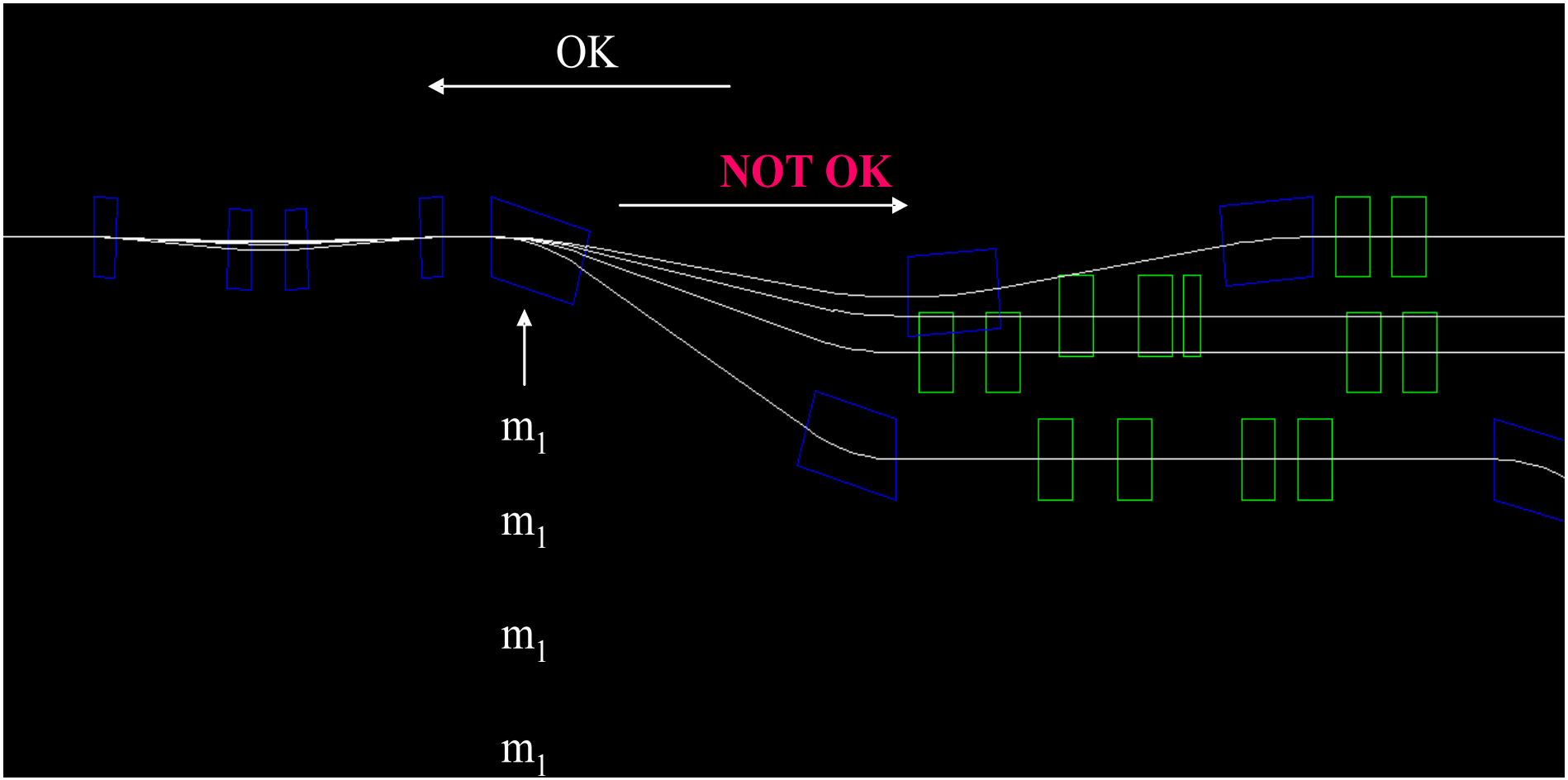
If m_1, m_{100}, m_{205}, and m_{315} are the same magnet, then they must be cloned if the recirculator is represented by an array

Is a beam line a (linked) list of magnet?

Actually no.....

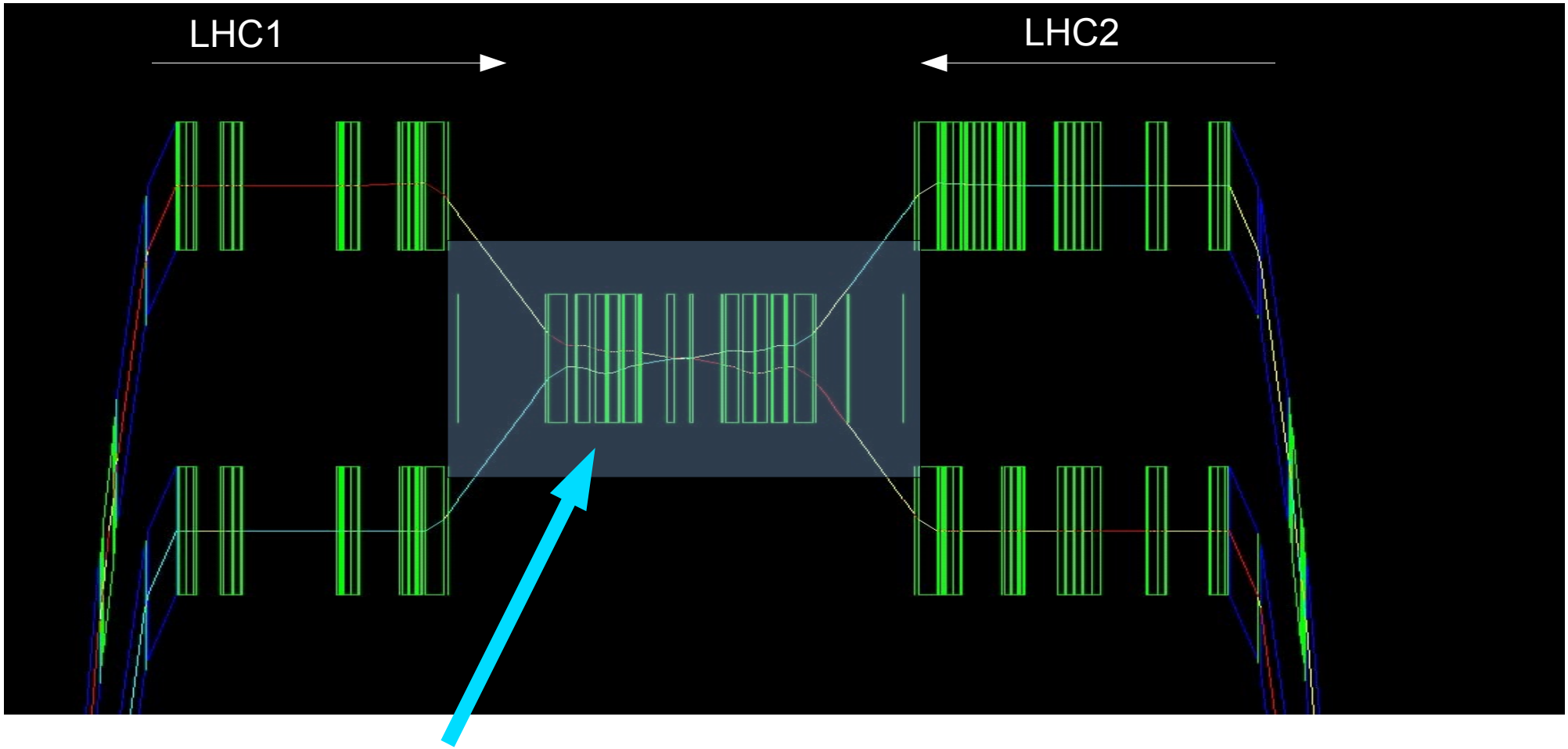
Let us examine a "recirculator" to focus our minds on the shortcomings of a simple list of magnets.

~~Linked List of Magnet~~



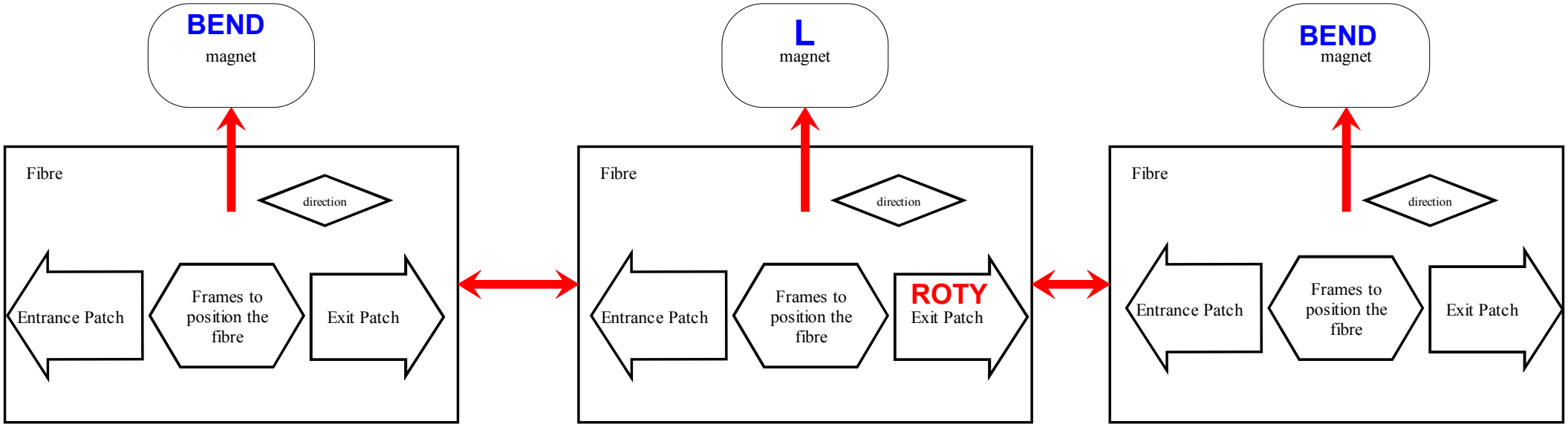
How about a collider, say LHC?

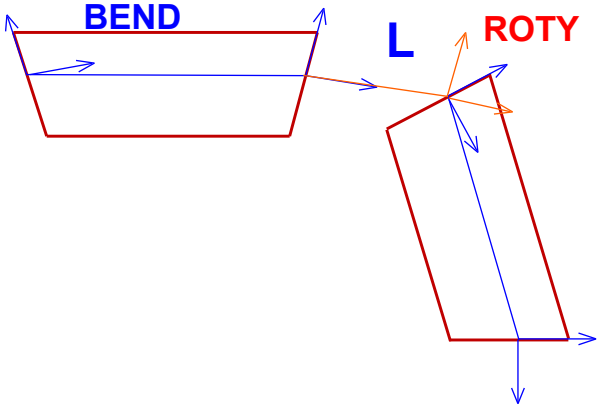
LHC Example: Intersecting rings



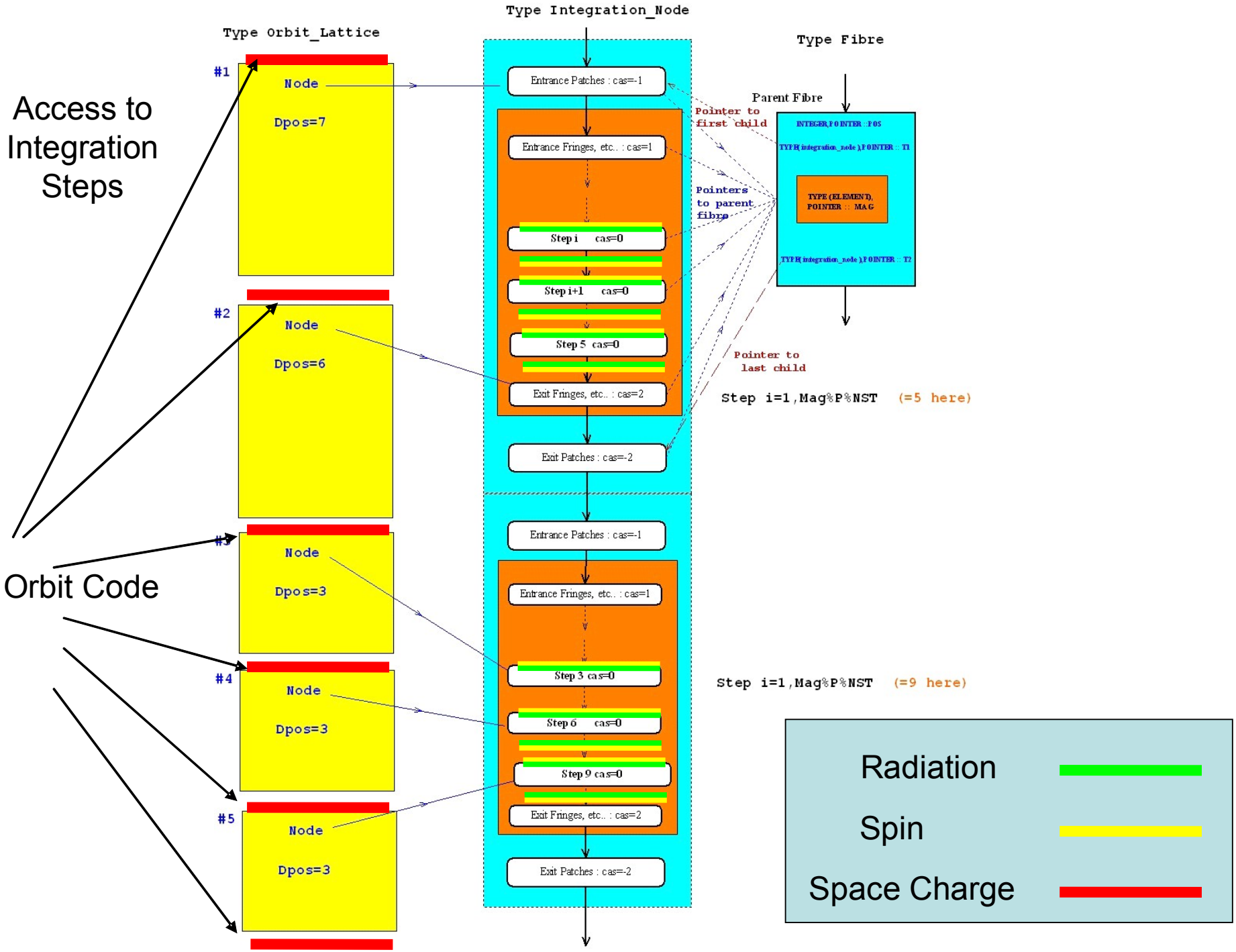
Interaction regions with common magnets in LHC1 and LHC2

My Solution: double linked list of "*Fibres*"





- ROTY (the rotation) is not a "magnet" anymore.
- Recirculation is not a problem because we do not come back to the same fibre. But it points to the same magnet!
- It solves the "collider" issue for the same reasons. Notice that the fibre has a propagation direction.


Type Integration_Node
Type Orbit_Lattice
Type Fibre
Access to Integration Steps
Orbit Code
#1
Node
Dpos=7
#2
Node
Dpos=6
#3
Node
Dpos=3
#4
Node
Dpos=3
#5
Node
Dpos=3
Entrance Patches : cas=-1
Entrance Fringes, etc.. : cas=1
Step i cas=0
Step i+1 cas=0
Step 5 cas=0
Exit Fringes, etc.. : cas=2
Exit Patches : cas=-2
Entrance Patches : cas=-1
Entrance Fringes, etc.. : cas=1
Step 3 cas=0
Step 6 cas=0
Step 9 cas=0
Exit Fringes, etc.. : cas=2
Exit Patches : cas=-2
Parent Fibre
Pointer to first child
Pointers to parent fibre
Pointer to last child
INTEGER,POINTER :: POS
TYPE(integration_node),POINTER :: T1
TYPE (ELEMENT), POINTER :: MAG
TYPE(integration_node),POINTER :: T2
Step i=1,Mag%P%NST (=5 here)
Step i=1,Mag%P%NST (=9 here)
Radiation
Spin
Space Charge